Capítulo 12: Autômatos

O nosso modelo de referência para computação mecânica é a máquina de Turing. Uma máquina Turing tem apenas dois componentes, uma CPU e uma fita. Vamos agora tirar a fita e estudar a CPU sozinha.

Dito de outra maneira, enquanto uma máquina de Turing tem memória ilimitada, os dispositivos que usamos todos os dias não têm. Podemos perguntar que tarefas podem ser feitas por uma máquina com memória limitada.

12.1. Máquinas de estados finitos

Produzimos um novo modelo de computação, modificando a definição da Máquina de Turing. Eliminaremos a capacidade de escrever, alterando a cabeça da fita de leitura/escrita para somente leitura. Isto dar-nos-á uma visão do que pode ser feito apenas com estados. Descobriremos que este tipo de máquina pode fazer muitas coisas, mas não tantas como uma máquina Turing.

No nosso novo modelo, utilizaremos o mesmo tipo de tabelas de transição e grafos de transição que fizemos com as máquinas de Turing.

Exemplo 12.1.1

Um interruptor de energia tem dois estados, q_{off} e q_{on} e o seu alfabeto de entrada tem um símbolo, `toggle`. (Figura 12.1)

Figura 12.1: Um interruptor de energia.

Exemplo 12.1.2

Opere esta catraca, colocando duas fichas (token) e depois passando (push) por ela. Ela possui três estados e o seu alfabeto de entrada é Σ = { `token`, `push` }.

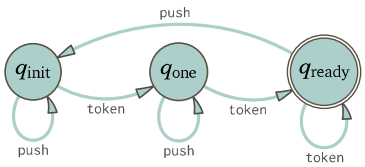


Figura 12.2: Uma catraca que deve receber duas fichas antes de liberar a passagem.

Como vimos com as máquinas de Turing, os estados são uma forma limitada de memória. Por exemplo, q_{one} é, no exemplo acima, como a catraca "se lembra" de ter recebido até agora uma ficha.

Exemplo 12.1.3

Esta máquina de venda automática dispensa artigos que custam 30 centavos. Ela aceita apenas moedas de 5, 10 e 25 centavos. A figura é complexa, por isso vamos mostrá-la em três camadas. A primeira é das setas para as moedas de 5 centavos (que chamamos de *n*, de *nickel* em inglês) e o apertar do botão de distribuição.

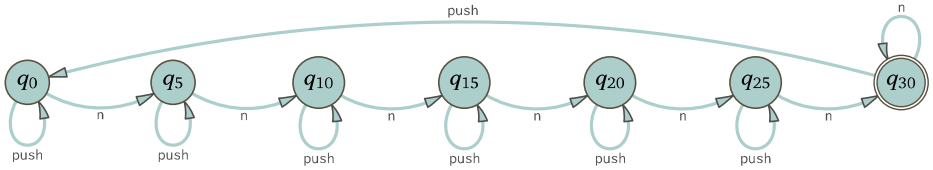


Figura 12.3: Máquina de vendas. Camada para as moedas de 5 centavos.

Após receber 30 centavos e receber outra moeda de 5 centavos, esta máquina faz algo não muito sensato: fica em q_{30}. Na prática, uma máquina teria mais estados para acompanhar os excessos, para que pudéssemos dar troco, mas aqui ignoramos isso. A seguir vêm as setas para as moedas de 10 centavos (que chamamos de *d*, de *dime* em inglês)

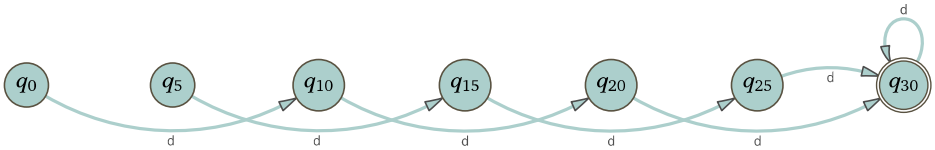


Figura 12.4: Máquina de vendas. Camada para as moedas de 10 centavos.

e para as moedas de 25 centavos (que chamamos de *q*, de *quarter* em inglês).

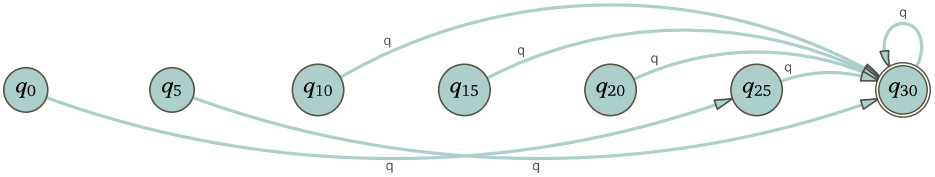


Figura 12.5: Máquina de vendas. Camada para as moedas de 25 centavos.

Exemplo 12.1.4

Esta máquina, quando iniciada no estado q_0 e alimentada com cadeias de bits, manterá o registro do resto módulo 4 da quantidade de 1's.

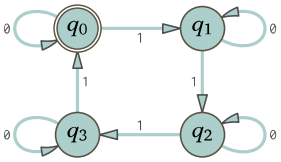


Figura 12.6: Quantidade de 1's módulo 4

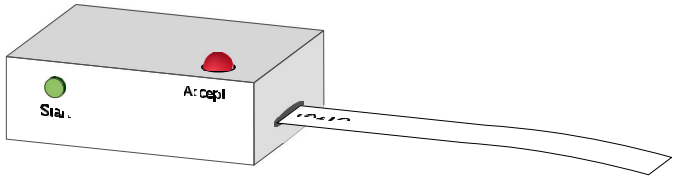
Definição

Uma **máquina de estados finitos**, ou **autômato de estados finitos**, é composta por cinco coisas, $\mathcal{M} = \langle Q, q_{\text{start}}, F, \Sigma, \Delta \rangle$. São elas: um **conjunto finito de estados** Q , um dos quais é o **estado inicial** q_{start}, um subconjunto $F \subseteq Q$ de **estados de aceitação** ou **estados finais**, um conjunto **alfabeto de entrada** finito Σ, e uma **função do próximo estado** ou **função de transição** $\Delta : Q \times \Sigma \to Q$.

Isto pode não parecer imediatamente como a nossa definição de uma Máquina de Turing. Parte disso deve-se ao fato de já termos definido os termos 'alfabeto' e 'função de transição'. As outras diferenças decorrem do fato de que as máquinas de estados finitos não podem escrever. Em primeiro lugar, por não poderem escrever, as máquinas de estados finitos não precisam mover a fita para o trabalho de rascunho, por isso abandonamos os símbolos de ações de fita L e R.

A outra diferença entre as máquinas de estados finitos e as máquinas de Turing é a presença dos estados de aceitação. Considere, na máquina de venda automática do Exemplo 12.1.3, o estado q_{30}. Ele é um estado de aceitação, o que significa que a máquina viu na entrada o que procura. O mesmo se aplica ao estado q_{ready} do Exemplo 12.1.2 e ao estado q_{on} do interruptor de alimentação do Exemplo 12.1.1. Embora possamos conceber uma Máquina de Turing para indicar uma escolha, organizando de modo que, para cada entrada, a máquina parará e a única coisa na fita será um 1 ou 0, uma máquina de estados finitos fornece uma decisão, terminando num destes estados designados. Podemos imaginar que os estados de aceitação são ligados a uma luz vermelha para que saibamos quando uma computação é bem-sucedida. Nos grafos de transição denota-se os estados finais com círculos duplos e nas tabelas de funções de transição marcamo-los com um '+'.

Para trabalhar com uma máquina de estados finitos, colocamos a entrada de comprimento finito na fita e pressionamos Start.



A máquina consome a entrada, em cada passo apagando o caractere anterior da fita e depois lendo o seguinte. Podemos rastrear os passos quando a máquina de módulo 4 do Exemplo 12.1.4 recebe a entrada `10110`.

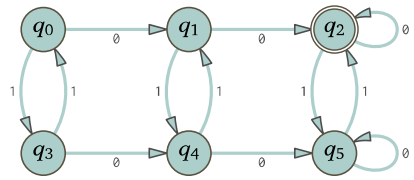
Passo	Configuração
0	1 0 1 1 0 q_0
1	0 1 1 0 q_1
2	1 1 0 q_1
3	1 0 q_2
4	0 q_3
5	q_3

Consequentemente, não há problema da parada para as máquinas de estados finitos — elas sempre param após um número de passos igual ao comprimento da entrada.

No final, ou a luz `Accept` está acesa ou não está. Se estiver acesa, então dizemos que a máquina **aceita** a cadeia de entrada, caso contrário **rejeita** a cadeia.

Exemplo 12.1.5

Esta máquina aceita uma string se e somente se contiver pelo menos dois `0`'s, bem como um número par de `1`'s. (O '+' ao lado de q_2 marca-o como um estado de aceitação).

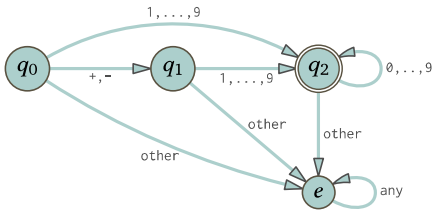


Δ	`0`	`1`
q_0	q_1	q_3
q_1	q_2	q_4
+ q_2	q_2	q_5
q_3	q_4	q_0
q_4	q_5	q_1
q_5	q_5	q_2

Esta máquina ilustra o segredo para conceber máquinas de estado finito, que cada estado tenha um significado intuitivo. O estado q_4 significa "até agora, a máquina viu um 0 e um número ímpar de 1's". E q_5 significa "até agora, a máquina viu dois 0's mas um número ímpar de 1's". O grafo traz à tona este princípio. A sua primeira linha tem estados que até agora têm visto um número par de 1's, enquanto os estados da segunda linha viram um número ímpar. A sua primeira coluna contém estados que não viram 0's, a segunda coluna contém estados que viram um zero, e a terceira coluna tem estados que viram dois 0's.

Exemplo 12.1.6

Esta máquina aceita strings que são válidas como representações decimais de números inteiros. Assim, ela aceita '21' e '−707' mas não aceita '501−'. Tanto o grafo de transição como a tabela agrupam algumas entradas quando resultam na mesma ação. Por exemplo, quando no estado q_0 esta máquina faz a mesma coisa, quer a entrada seja + ou −, ou seja, passa para q_1.



Δ	`+, -`	`0, ... 9`	*else*
q_0	q_1	q_2	e
q_1	e	q_2	e
+ q_2	e	q_2	e
e	e	e	e

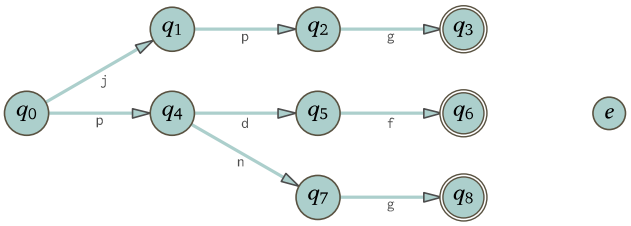
Qualquer caractere de entrada incorreto envia a máquina para o estado de erro, e, que é um estado de sumidouro, o que significa que a máquina nunca sai desse estado.

As nossas descri]ções de máquina de estado finito assumirão geralmente que o alfabeto é claro a partir do contexto. Por exemplo, o exemplo anterior apenas diz "else" (ou no grafo, "other"). Na prática, tomamos o alfabeto como sendo o conjunto de caracteres que alguém poderia possivelmente digitar, incluindo letras como `a` e `A` ou caracteres como ponto de exclamação ou abre parênteses. Assim, a concepção de uma

máquina de estados finitos de acordo com um padrão moderno poderia utilizar todo o Unicode. Mas para os exemplos e exercícios aqui, vamos utilizar pequenos alfabetos.

Exemplo 12.1.7

Esta máquina aceita cadeias que são membros do conjunto { `jpg`, `pdf`, `png` } de extensões de nomes de arquivo. Note que ela tem mais de um estado final.

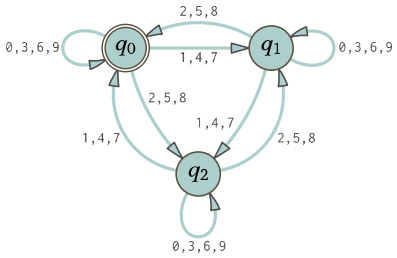


Esse desenho omite muitas arestas, as que envolvem o estado de erro e. Por exemplo, do estado q_0 qualquer caractere de entrada que não seja `j` ou `p` é um erro. (Colocar todas as arestas faria uma bagunça. Casos como este são aqueles em que a tabela de transição é melhor do que a figura do grafo. Mas a maioria das nossas máquinas são pequenas e têm apenas alguns estados e arestas, por isso, normalmente preferimos a figura).

Este exemplo ilustra que para qualquer linguagem finita existe uma máquina de estado finito que aceita uma cadeia se e somente se for um membro da linguagem. A ideia é colocar as cadeias em ordem alfabética, e para aquelas que têm prefixos comuns, a máquina passa pelas partes partilhadas em conjunto, como aqui com `pdf` e `png`.

Exemplo 12.1.8

Embora não tenham memória de rascunho, as máquinas de estados finitos podem realizar trabalhos úteis, como alguns tipos de aritmética. Esta máquina aceita cadeias representando um número natural que é um múltiplo de três, tais como `15` e `5013`.

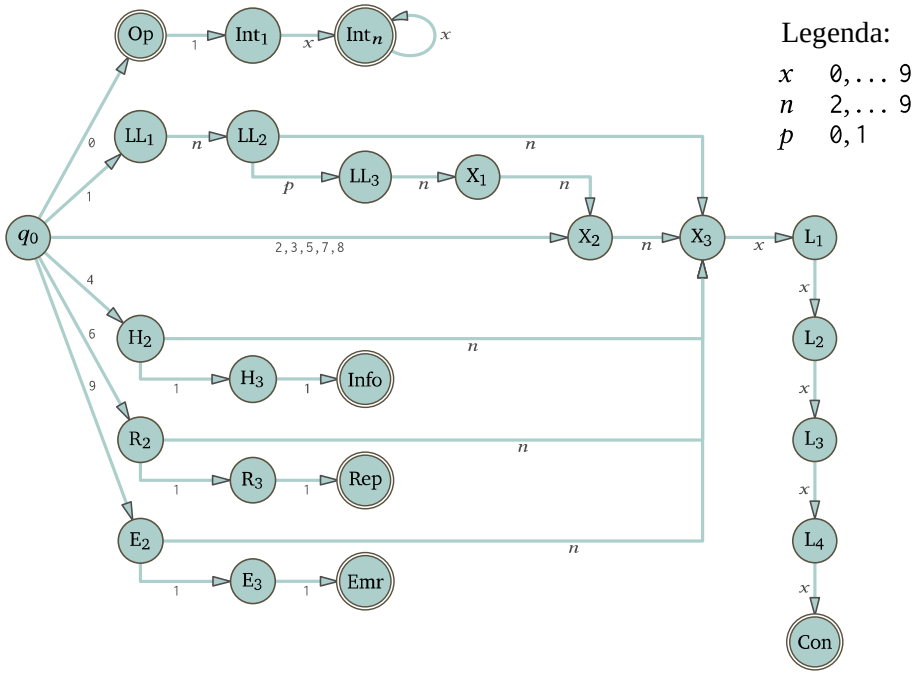


Como q_0 é um estado de aceitação, esta máquina aceita a cadeia vazia.

Exemplo 12.1.9

Esta é uma versão simplificada de como os números de telefone costumavam ser tratados na América do Norte. Considere o número 1-802-555-0101. O 1 inicial significa que a chamada deve deixar a central local para ir para as linhas de longa distância. O 802 é um código de área; o sistema pode perceber isso porque o seu segundo dígito é 0 ou 1, portanto não é uma central local da mesma área. A seguir o sistema processa o número de central local, o 555, encaminhando a chamada para uma central

local física específica. Essa central processa o número de linha 0101, e faz a conexão.



Hoje em dia, o quadro é muito mais complicado. Por exemplo, já não é necessário que os códigos de área tenham um dígito central 0 ou 1. Esta complicação adicional é possível porque em vez de comutar com dispositivos físicos, fazemo-lo agora em software.

Após a definição da máquina de Turing, demos uma descrição formal da ação dessas máquinas. A seguir, fazemos o mesmo aqui.

Uma **configuração** de uma máquina de estados finitos é um par $C = \langle q, \tau \rangle$, onde q é um estado, $q \in Q$, e τ é uma cadeia (possivelmente vazia), $\tau \in \Sigma^*$. Iniciamos uma máquina com alguma cadeia de **entrada** τ_0 e dizemos que a configuração inicial é $C_0 = \langle q_0 , \tau_0 \rangle$.

Uma máquina de estados finitos atua por uma sequência de **transições** de uma configuração para outra. Para $s \in \mathbb{N}^+$ a configuração da máquina após a s-ésima transição é a sua configuração no **passo** s , C_s .

Esta é a regra para fazer uma transição (dizemos por vezes que é uma transição **permitida** ou **legal**, para enfatizar). Suponha que a máquina está na configuração $C_s = \langle q, \tau_s \rangle$. No caso de τ_s não ser vazia, apague o símbolo inicial da cadeia, c. Ou seja, onde $c = \tau_s[0]$, pegue $\tau_{s+1} = \langle \tau_s[1], \ldots \tau_s[k] \rangle$ para $k = |\tau_s| - 1$. Então, o próximo estado da máquina é $\hat{q} = \Delta(q, c)$ e a sua próxima configuração é $C_{s+1} = \langle \hat{q}, \tau_{s+1} \rangle$. Denote esta relação antes-depois entre as configurações por $C_s \vdash C_{s+1}$.[1]

O outro caso é quando a string τ_s é vazia. Esta é a **configuração de parada** C_h. Nenhuma transição se segue a partir da configuração de parada.

1 Assim como para as Máquinas de Turing, leia o símbolo $\vdash$ como “deriva em um passo”.

A cada transição, o comprimento da string da fita diminui de um, de modo que cada computação eventualmente atinge uma configuração de parada $C_h = \langle q, \varepsilon \rangle$. Uma computação de máquina de estados finitos é uma sequência $C_0 \vdash C_1 \vdash C_2 \vdash \ldots \vdash C_h$. Podemos abreviar essa sequência por $\vdash^*$, como em $C_0 \vdash^* C_h$.[2]

Se o estado ao se chegar na configuração de parada for um estado final, $q \in F$, então a máquina aceita a entrada τ_0, caso contrário ela rejeita τ_0.

Note que, tal como no formalismo para as máquinas de Turing, o cerne das definições é a função de transição Δ. Ela faz a máquina mover-se passo a passo, de configuração em configuração, em resposta à entrada.

Exemplo 12.1.10

A máquina de múltiplos de três do Exemplo 12.1.8 fornece a computação. $\langle q_0$, 5013$\rangle$ $\vdash \langle q_2$, 013$\rangle \vdash \langle q_2$, 13$\rangle \vdash \langle q_0$, 3$\rangle \vdash \langle q_0, \varepsilon \rangle$. Uma vez que q_0 é um estado de aceitação, a máquina aceita 5013.

Definição

O conjunto de cadeias aceitas por uma máquina de estados finitos $\mathcal{M}$ é a **linguagem dessa máquina, $\mathcal{L}(\mathcal{M})$**, ou a linguagem **reconhecida**, ou **decidida**, (ou **aceita**), pela máquina.

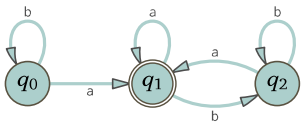
Para as máquinas de estados finitos, decidir uma linguagem é equivalente a reconhecê-la. 'Reconhecer' é o termo mais comum do que 'Decidir' aqui.

Definição

Para qualquer máquina de estados finitos com função de transição $\Delta : Q \times \Sigma \to Q$, a **função de transição estendida** $\hat{\Delta} : \Sigma^* \to Q$ fornece o estado em que a máquina terminará após ter começado no estado inicial e consumido a cadeia dada.

Exemplo 12.1.11

A função de transição estendida $\hat{\Delta}$ desta máquina



Δ	a	b
q_0	q_1	q_0
+ q_1	q_1	q_2
q_2	q_1	q_2

estende a sua função de transição ordinária Δ na medida em que repete a primeira linha da tabela de Δ.

$$\hat{\Delta}(\mathsf{a}) = q_1 \quad \hat{\Delta}(\mathsf{b}) = q_0$$

2 Leia o símbolo $\vdash^*$ como "deriva eventualmente", ou simplesmente "deriva".

(Ignoramos a diferença entre os caracteres de entrada Δ e as cadeias de caracteres de comprimento um de $\hat{\Delta}$). Aqui está o efeito de $\hat{\Delta}$ nas cadeias de caractere de comprimento dois.

$$\hat{\Delta}(\mathtt{aa}) = q_1 \quad \hat{\Delta}(\mathtt{ab}) = q_2 \quad \hat{\Delta}(\mathtt{ba}) = q_1 \quad \hat{\Delta}(\mathtt{bb}) = q_0$$

Note que isso requer determinismo, sem o qual $\hat{\Delta}$ não estaria bem definida; Δ tem um estado seguinte para todas as configurações de entrada e assim, por indução, para todas as cadeias de entrada, $\hat{\Delta}$ tem um estado de saída ao final.

Finalmente, note a semelhança entre $\hat{\Delta}$ e ϕ_e, a função computada pela máquina de Turing $\mathcal{P}_e$. Ambas tomam como entrada o conteúdo da fita inicial da sua máquina, e ambas fornecem como saída o resultado da sua máquina.

12.2. Não-determinismo

FIXME TODO Continua... Baixe novamente quando for atualizado

12.3. Créditos

Todas as seções, foram adaptadas (traduzidas e modificadas) de [1], que está disponível sob a licença Creative Commons Attribution-ShareAlike 4.0 (CC BY-SA 4.0).

12.4. Referências

1. Hefferon, Jim. *Theory of Computation: Making Connections*. Disponível em http://hefferon.net/computation/

12.5. Licença