

LINGUAGENS FORMAIS E AUTÔMATOS

CONCEITOS INTRODUTÓRIOS SOBRE LINGUAGENS FORMAIS E AUTÔMATOS

Autor: Me. Pedro Henrique Chagas Freitas

Revisor: Edilson José Rodrigues

Tempo de leitura do conteúdo estimado em 1 hora.

Introdução

Olá, caro(a) estudante! É com entusiasmo que convido você para a leitura deste material. Você sabia que, em 1950, nasceu a teoria das **linguagens formais**, com o objetivo de tratar as **linguagens naturais** para desenvolver as **linguagens computacionais**? Desde o início, foi verificada a importância dos estudos das linguagens formais, com ênfase na análise léxica e sintática, até a formação dos hipertextos, hipermídias e linguagens não lineares. A partir disso, surgiram os estudos aplicados às estruturas das linguagens, que deram origem aos autômatos, fundamentais para o desenvolvimento das linguagens computacionais que temos hoje.

Diante disso, a teoria por trás das linguagens formais e dos autômatos foi um marco no desenvolvimento científico computacional, representando uma mudança transformadora na forma como as linguagens computacionais são pensadas. Por fim, vale mencionar que várias das temáticas aqui abordadas não se restringem ao campo prático, ou seja, tratam-se, muitas vezes, de inferências matemáticas que, por sua vez, deram origem à teoria dos conjuntos, à teoria do bombeamento para linguagens regulares, ao próprio conceito de autômato, às gramáticas regulares, à hierarquia de Chomsky, dentre outros.

Com essa breve introdução, garanto que o assunto é extremamente interessante e meu papel será apresentar a temática “linguagens formais e autômatos”, da forma mais simples e didática possível.

Boa leitura!

Introdução e Conceitos Básicos: Sintaxe, Semântica, Conjuntos, Relações, Alfabeto e Funções

Caro(a) estudante, é de seu conhecimento que a teoria das linguagens formais foi desenvolvida em meados da década de 1950 com o objetivo de implementar teorias relacionadas com as linguagens naturais?

Entretanto, logo foi visto que a teoria das linguagens formais era fundamental para o que alguns chamavam de linguagens artificiais, em especial, para as linguagens originárias da computação. Desde essa época, o estudo das linguagens formais apresenta vários enfoques, com destaque para as aplicações em análise léxica e em análise sintática das linguagens de programação e da modelagem de circuitos lógicos.

Historicamente, no estudo das linguagens de programação, o

problema sintático foi reconhecido antes do problema semântico e foi o primeiro a receber um tratamento adequado, tendo em vista que os problemas sintáticos são de tratamentos mais simples que os semânticos. Como consequência, foi dada uma grande ênfase à sintaxe, ao ponto de levar à ideia de que questões das linguagens de programação resumiam-se às questões de sintaxe. Atualmente, a teoria da sintaxe possui construções matemáticas bem definidas e universalmente reconhecidas como, por exemplo, as gramáticas de Chomsky (MENEZES, 2010, p. 19).

Um dos principais campos, basilar e fundamental das linguagens formais, a título de exemplo, são as linguagens formais aplicadas ao desenvolvimento *web* , tendo em vista que a *web* é, sem dúvidas, o maior, senão um dos maiores, passos da evolução já dados pela humanidade. Isso somente ocorreu tendo em vista o emprego dos estudos das linguagens formais e autômatos. Note que a programação *web* , ou seja, às linguagens de hipertexto, tem sua origem diretamente atrelada aos estudos das linguagens regulares, assim, a teoria das linguagens regulares e dos autômatos foi essencial na elaboração de soluções práticas, como a *web* .

*E é agora que você, estudante,
pode se perguntar: Quão gerais
são as linguagens regulares?
Seria toda linguagem formal
regular?*

A primeira questão que levantaremos é sobre o que acontece quando operamos linguagens regulares. No caso, as operações que mencionamos são tais como matemáticas, união e concatenação, herdadas da teoria matemática dos conjuntos.

A segunda questão que precisamos levantar trata da habilidade presente na linguagem humana para decidir sobre certas propriedades. Por exemplo, podemos fazer um algoritmo que decida se uma linguagem regular é finita ou não? Diante disso, fica demonstrado que as linguagens naturais humanas, seja qual for o idioma, sempre serão o ponto de partida para o estudo das linguagens formais e dos autômatos; por essa razão, vale destacar que não estamos criando a roda nem reinventando-a, estamos aplicando propriedades naturais, em conjunto com teorias matemáticas, para demonstrar a aplicabilidade do funcionamento das linguagens, que, no caso, são as linguagens formais e os autômatos.

Esclarece-se, então, que considerar se uma dada linguagem é regular ou não é uma tarefa que permeia entender o que faz uma linguagem ser regular, isto é, compreender os fenômenos da sintaxe, da semântica, dos conjuntos, das relações, do alfabeto, das gramáticas e das funções é essencial no trato das linguagens formais e dos autômatos.

Podemos sempre demonstrar isso exibindo uma equação $f(x)$ que representa uma linguagem formal, ou uma expressão regular que a denota, ou uma gramática regular que impõe restrições para o desenvolvimento da linguagem.

Logo, conforme veremos em alguns momentos, podemos não visualizar a função que reconhece a linguagem regular, porém, isso não significa necessariamente que não exista um autômato correspondente. Por essa razão, estudamos as propriedades gerais das linguagens regulares e dos autômatos.

Caro(a) estudante, para somar ainda mais conhecimento ao campo de

estudos que estamos nos debruçando, vamos verificar o quadro a seguir:

Visão formal das linguagens de programação

Uma entidade livre, sem nenhum significado associado.

Uma entidade dotada de significado.

Quadro 1.1 - Visão formal quanto ao significado de linguagem de programação, isto é, como uma entidade livre sem interpretação ou com interpretação.

Fonte: Elaborado pelo autor.

#PraCegoVer : a imagem apresenta o Quadro 1.1, dividido em três linhas. Seguindo a ordem de cima para baixo, temos, na primeira linha, o título do quadro: “Visão formal das linguagens de programação”. Na segunda linha, temos: “Uma entidade livre, ou seja, sem qualquer significado associado”. Na terceira e última linha, temos: “Uma entidade dotada de significado.”

A sintaxe é responsável pelas propriedades livres das linguagens, dentre elas, temos a verificação gramatical dos *softwares* . Já a semântica tem por objetivo realizar a interpretação da sintaxe, entregando, assim, um significado para um determinado *software* . Consequentemente, a sintaxe basicamente converte símbolos, desconsiderando os seus correspondentes significados.

É importante salientar que a resolução de qualquer problema real ocorre a partir da interpretação semântica dos símbolos, por exemplo, o símbolo *W* representa uma variável ou uma constante, um número *float* ou um inteiro.

Dessa forma, não existe uma concepção de programa “errado”. Ou seja, podemos ter, de um lado, um programa sintaticamente válido, isto é, “correto”, porém, sem apresentar o que o programador esperava quando implementou o respectivo código. Assim, a decisão de considerar um programa “correto” ou “errado” deve ser balizada pelo comportamento esperado (MENEZES, 2010).

Observe que não são todas as vezes que ficam claras as diferenças entre sintaxe e semântica. Vamos a um exemplo para entender melhor?

A presença de um nome em um programa pode ser tratado de forma simples, tendo em vista um problema sintático ou semântico. Diante disso, podemos concluir que a análise léxica poderá ser vista como um tipo especial de análise sintática, a qual é centrada nos componentes básicos da linguagem. Dessa forma, linguagens formais também se preocupam com os problemas léxicos. Precisamos nos aprofundar mais nesse assunto, concorda? Vamos lá!

Abordagem das linguagens formais

Operacional

Axiomático

Denotacional

Linguagens de
programação

#PraCegoVer : o infográfico, em forma de figura com desenho ao fundo, apresenta abas para interagir, clicar e visualizar cada conteúdo. O infográfico tem o título geral “Abordagem das linguagens formais”, e o desenho é composto por

quatro ícones, sendo eles: a tela de um computador com a palavra “Operacional” ao lado, que leva ao segundo ícone, uma lupa, com a palavra “Axiomático”, que leva ao terceiro ícone, um cérebro com a palavra “Denotacional”, que leva ao quarto e último ícone, uma lâmpada com as palavras “Linguagens de programação”. As abas na lateral possuem os respectivos conteúdos: 1º “Operacional: define-se uma linguagem de máquina abstrata, com base em estados, ou seja, em instruções primitivas e na especificação de como cada instrução modifica cada estado”; 2º “Axiomático: regras de busca são associadas aos componentes da linguagem. As regras permitem afirmar o que é verdadeiro e o que é falso após a ocorrência de cada uma delas”; 3º “Denotacional: também conhecido como formalismo funcional, é definido quando temos um domínio que permite a caracterização do conjunto de palavras admissíveis na linguagem, logo, temos uma construção específica de valores e de subcomponentes”; 4º “Linguagens de programação: mecanismo de implementação de comportamentos computacionais, mediante código, isto é, mediante linguagem previamente definida, com características próprias e interpretativa por parte de um compilador”.

Agora que viu o infográfico e ficou mais por dentro das linguagens formais, podemos debater o respectivo problema fundamental: dada uma linguagem arbitrária L e uma cadeia w , é possível determinar algoritmicamente se w pertence a L ? Essa questão é tratada como questão de pertinência, e o método para respondê-la é o algoritmo de pertinência.

A questão da existência e do respectivo algoritmo trata da pertinência como um problema em geral, difícil de ser tratado. Porém, para as linguagens regulares, temos uma questão simples, considerando que a pertinência é fundamental para elas. Vejamos:

O que queremos dizer quando falamos de alguma linguagem? Estamos dizendo que temos diversas maneiras de descrever as linguagens regulares, como descrição verbal informal, notações de conjunto, autômatos finitos, expressões regulares e gramáticas regulares. Porém, somente as três últimas são suficientemente bem definidas para serem usadas nos teoremas que

veremos, dada a sua definição algébrica.

Portanto, dizemos que uma linguagem regular é dada em uma representação padrão se, e somente se, ela for descrita por um autômato finito, ou como uma expressão regular ou como uma gramática regular.

Quanto à teoria dos conjuntos, é fundamental para o estudo de linguagens formais e autônomas, pois todos os conceitos desenvolvidos, bem como os correspondentes produtos ou resultados, têm por base os conjuntos ou as construções sobre conjuntos.

Conjunto é uma estrutura responsável por agrupar objetos e constitui um fundamento na construção de estruturas mais complexas. Logo, um conjunto é uma coleção, sem repetições e sem qualquer ordenação, de objetos denominados elementos (MENEZES, 2010).

Um elemento, por sua vez, é um objeto concreto ou abstrato, assim, temos que o elemento é uma entidade básica em um conjunto. Conclui-se que um conjunto pode ser definido como a coleção de zero ou mais objetos distintos.

Já as relações dizem respeito ao conceito usado com frequência ao longo de todo texto de uma linguagem, ou seja, é a base para o correto entendimento das construções matemáticas presentes nas linguagens. Por sua vez, são as relações que nascem as funções das linguagens. No caso dos estudos das linguagens formais e autômatos, os conceitos de relação estão interligados com ordem e equivalência.

As funções, por conseguinte, são divididas em: função parcial e total. Uma função parcial é uma relação dentro da linguagem, na qual cada um dos elementos presentes no domínio está relacionado com, no máximo, um elemento externo ao domínio. Já uma função total é uma função definida para todos os elementos de um domínio.

Os conceitos estudados abrem caminho para a estruturação das linguagens formais e dos autômatos através do alfabeto e das palavras. Obviamente, a abstração de uma palavra é uma cadeia de caracteres acompanhada de um

significado, logo, o alfabeto, isto é, a composição de uma palavra, é uma entidade abstrata básica da linguagem, não possuindo definição e formas; assim, letras e dígitos são exemplos de símbolos que compõem um alfabeto. Podemos, então, definir que um alfabeto é um conjunto finito de símbolos ou de caracteres.

Por sua vez, a palavra ou a cadeia de caracteres ou a sentença é uma sequência de característica finita de símbolos do alfabeto da linguagem, que reunidos formam um significado. Logo, uma cadeia de símbolos só é uma palavra, dentro de uma linguagem, se apresentar um significado ou derivar uma função; a simples concatenação de símbolos de um alfabeto não forma, por si só, uma palavra.

Por fim, em uma linguagem formal, o conjunto finito de todas as palavras forma a gramática da linguagem. A gramática é a definição das regras que, quando aplicadas, geram as palavras. Assim, um conjunto de todas as palavras geradas por uma gramática dá origem a uma linguagem formal.

Conhecimento

Teste seus Conhecimentos

(Atividade não pontuada)

Um conjunto é uma estrutura que agrupa objetos e que constitui a base para construir estruturas mais complexas, como alfabetos e palavras. Dessa forma, podemos definir que um conjunto é uma coleção, sem repetições e sem qualquer ordenação, de objetos denominados elementos.

DIVERIO, T. A.; MENEZES, P. B. **Teoria da Computação** : máquinas universais e computabilidade. 3. ed. Porto Alegre: Grupo A, 2011. v. 5.

Diante das definições apresentadas sobre a formação dos conjuntos, assinale a alternativa que representa a relação em que cada elemento de um domínio está relacionado com, no máximo, um elemento externo a ele.

- ☐ a) Função total.
- ☐ b) Alfabeto.
- ☐ c) Palavra.
- ☐ d) Função parcial.
- ☐ e) Gramática.

Autômatos Finitos Determinísticos (AFD)

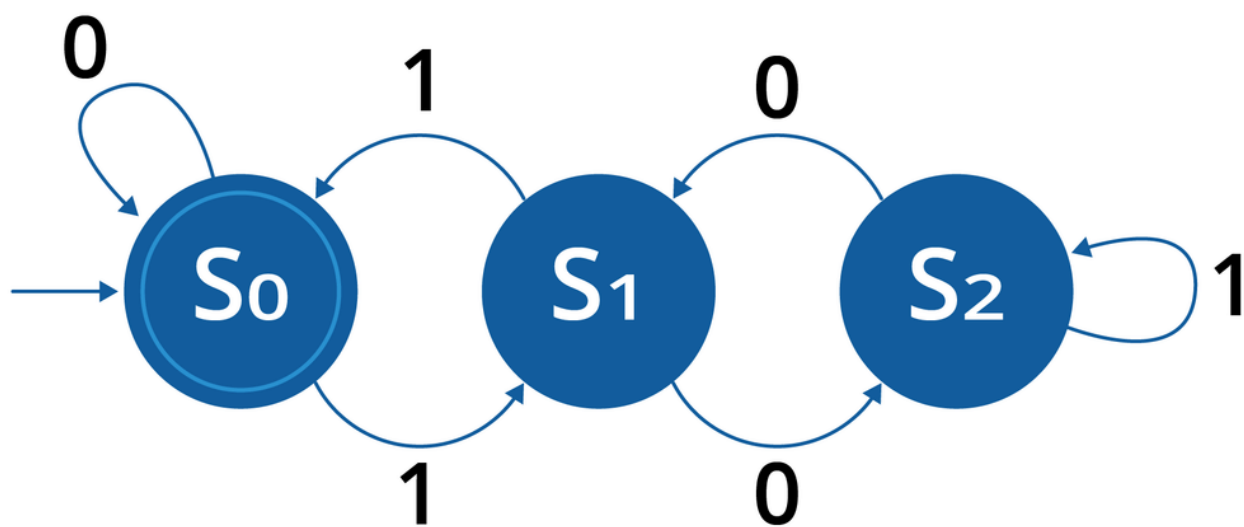
Caro(a) estudante, você sabia que os autômatos (finitos determinísticos, finitos não determinísticos e finitos com movimentos vazios) representam a

base do raciocínio típico das linguagens formais?

Basicamente, criou-se, a partir dos estudos sobre linguagens regulares, a teoria dos autômatos, que é um tópico basilar da ciência da computação, ou seja, os estudos sobre autômatos deram origem a muitas máquinas consideradas pré-computadores; logo, os próprios computadores se beneficiaram, no ciclo de evolução da computação, do emprego e dos estudos dos autômatos.

A palavra autômato é uma palavra de origem grega, que significa “agindo pela vontade própria”. Um exemplo comum seria um relógio de parede cuco, que é uma máquina que se move sem ajuda de eletricidade, ou seja, como um autômato.

Os autômatos finitos determinísticos (AFD) são máquinas de estados finitos, ou seja, aceitam ou rejeitam cadeias de símbolos, criando ramos únicos a partir de cada entrada. Para que você possa compreender melhor, veja a figura a seguir.



© VG Educacional

Figura 1.1 - Ilustração através de um diagrama de transição de estados de um autômato finito determinístico (AFD)

Fonte: Elaborada pelo autor.

#PraCegoVer : temos, na figura, a transição de estados, através de um diagrama, no qual é representado um autômato finito determinístico, com seus respectivos estados: 0 ou 1, nos momentos de inicialização (s_0); e os demais consecutivos, isto é, s_1 e s_2 , demonstrando, assim, a transição de estados finita e determinada.

Como você pôde ver na figura, temos um exemplo de autômato finito determinístico que trabalha com a recepção de números binários (0 e 1). No caso, S_0 , S_1 e S_2 são os estados do autômato, por isso, dizemos que um autômato é uma máquina de estados. Por fim, vale mencionar que o S_0 é o estado inicial, ou seja, é no S_0 que temos a verificação de aceitação do dado por parte do autômato.

Mas, por que dizemos que um autômato é determinístico? Como demonstrado, temos uma máquina de estados que deriva uma unidade de processamento ou unicidade de processamento, assim, toda vez que um dado (número, símbolo, variável etc.) se apresenta ao autômato, deve passar

pelo crivo da aceitação, isto é, deve ser determinado se o dado será processado ou não.

Em 1943, tivemos o surgimento dos primeiros conceitos, ainda teóricos, sobre a aplicabilidade dos autômatos finitos, pelo professor McCulloch. Basicamente, na época, buscava-se reproduzir estruturas simples de criação de máquinas de estados finitos, o que, por sua vez, derivou a teoria dos autômatos.

Na maioria dos casos, quando vislumbramos um autômato finito determinístico, estamos, na verdade, visualizando um diagrama de transição de estados. No exemplo da Figura 1.1, temos um AFD (autômato finito determinístico) sendo representado por um diagrama de transição de estados, com três estados, como já mencionado, S0, S1 e S2.

Observe que, como se trata de um autômato com sequência finita, isto é, finitos estados, teremos só estados binários, no caso, 0 ou 1; assim, só serão possíveis na entrada sequências finitas de caracteres 1s e 0s. Em cada estado (S0, S1 ou S2), temos uma transição, representada pelos arcos, que, por sua vez, pode derivar um 0 ou um 1, no estado posterior.

Na prática, isso significa que, em um dado estado, após ser realizada a leitura de um caractere (0 ou 1), o autômato determinará a transição para o próximo estado, obviamente com base em alguma regra. Por exemplo, se o autômato estiver no estado S0 e o caractere de entrada for 1, então teremos uma transição para o estado S1; daqui nasce a regra de que todo autômato finito determinístico possui um estado inicial ou estado de aceitação.

Um autômato finito determinístico é um conceito matemático abstrato, por essa razão equações são empregadas para explicar seus comportamentos; todavia, no caso do autômato finito determinístico, temos o emprego prático na criação de *hardwares* e *softwares* que resolvem problemas específicos.

Uma maneira simples de ver como isso pode acontecer é com um servidor de *e-mail*, ou de correio eletrônico, em que, dado o envio de uma requisição a um servidor, é possível acessar um dado *e-mail*, abstraindo as questões de

autenticidade e os protocolos de comunicação; temos uma máquina de estados finitos, isto é, um autômato finito determinístico. Logo, imagine como seria a vida se não tivéssemos *e-mails* ? Pois bem, agora você sabe que só os temos porque antes deles nasceu a teoria matemática dos autômatos finitos determinísticos.

Vale destacar que os autômatos finitos determinísticos são empregados nas linguagens regulares a fim de implementar, no baixo nível de programação, o que chamamos, no alto nível, de lógica de programação. Assim, antes de um programador desenvolver um código que, de maneira lógica, realize alguma função, este código deve operar uma linguagem regular, que, por sua vez, emprega autômatos na sua essência.

Um exemplo clássico é a análise léxica e o reconhecimento de padrões, que, como veremos posteriormente, são o emprego dos autômatos em linguagens regulares. Veremos, também, que um autômato finito determinístico pode ser construído a partir de um autômato finito não determinístico; logo, ambos são fundamentais para o desenvolvimento e a evolução da computação como a conhecemos hoje.

Um autômato é então definido como um modelo matemático de uma máquina de estados finitos. É aí que surge uma pergunta; clique no elemento para saber sobre:

Porém, o que é uma máquina de estados finitos?

Uma máquina de estados finitos é um modelo matemático empregado na representação de *softwares* de computadores, circuitos lógicos etc. Dessa forma, vemos que a definição de máquina de estados finitos é concebida como uma máquina abstrata que deve estar em um de seus finitos estados.

Fonte: Monsit Jangariyawong / 123RF.

A máquina de estados finitos está em um estado por vez, chamado estado atual. Um estado é responsável por armazenar informações sobre o passado, isto é, reflete as alterações que ocorreram desde a entrada, no início do sistema, até o presente. Uma transição entre estados indica uma mudança e pode ser verificada por uma condição que precisa ser realizada para que a transição ocorra. Uma ação é a apresentação de uma atividade em um determinado momento.

Até agora, verificamos que uma máquina de estados finitos é um modelo matemático empregado para modelar problemas, projetar programas de computador e circuitos lógicos digitais, entre outros. Trata-se de uma máquina abstrata que possui um número finito de estados e diversas transições entre eles.

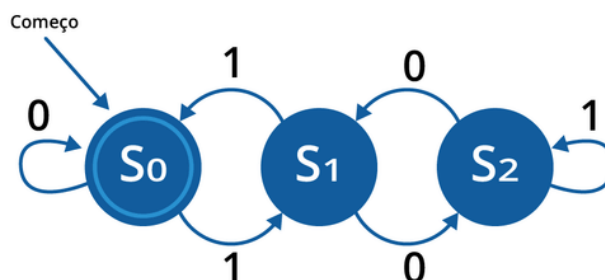
Você deve estar se perguntando, caro(a) estudante, por que utilizar essa abordagem? Ora, porque fica mais fácil de depurar (código modularizável), tendo em vista que, geralmente, não necessita de muito processamento; e porque se trata de um modelo com flexibilidade (novos estados e transições podem ser facilmente adicionados). Uma moto, no caso, pode ser modelada possuindo apenas dois estados: ligada ou desligada. Se eu ligar a ignição, a moto fica ligada; se eu desligar a ignição, há uma transição de estados e a moto é desligada (MENEZES, 2010).

Dessa forma, quando um autômato trabalha como um reconhecedor de uma determinada linguagem, busca orquestrar uma máquina. Podemos ver o emprego, por exemplo, em editores de texto para reconhecer padrões.

Dito isso, qual a diferença entre um autômato determinístico e um autômato não determinístico? Um autômato finito determinístico (AFD), ou máquina de estados finita determinística, é uma máquina de estados finitos que aceita ou rejeita cadeias de caracteres, gerando um único ramo de computação para

cada cadeia de entrada (MENEZES, 2010). Caro(a) estudante, verifique a figura a seguir para somar ainda mais conhecimento ao campo de estudos sobre o qual estamos nos debruçando.

Um Autômato Finito Determinístico (AFD) é definido por uma quintupla $M = (S, Q, d, q_0, F)$, em que:



- S: Alfabeto de símbolos finitos de entrada;
- Q: Conjunto finito de estados possíveis para M;
- d: Função transição ou função programa definida em $Q \times S \rightarrow Q$;
- q_0 : Estado inicial de M, sendo q_0 pertencente a Q;
- F: Conjunto de estados finais, tal que F faz parte de Q.

© VG Educacional

Figura 1.2 - Definição através de um diagrama de transição de estados da função de um autômato finito determinístico (AFD)

Fonte: Menezes (2010, p. 67).

#PraCegoVer : a imagem apresenta o quadro de uma figura, contendo um diagrama de transição de estados da função de um autômato finito determinístico (AFD), com o alfabeto de símbolos finitos de entrada (S), o conjunto finito de estados possíveis para o AFD (Q), a função transição (d), o estado inicial (q_0) e o conjunto de estados finais (F).

A imagem apresenta um autômato finito determinístico visualizado em um diagrama de transição de estados. Nesse autômato, temos três estados: S0, S1 e S2. A entrada é constituída por uma sequência finita de caracteres, no caso, 1's e 0's.

Em cada estado da máquina, há uma transição que leva para outro estado,

isto é, para ambos caracteres do alfabeto de entrada.

Isso significa que, em um dado estado, após a leitura de cada caractere, a máquina determinística transita de um estado para outro. Por exemplo, se o autômato no estado atual for S_0 e o símbolo de entrada para aquela instância for igual a '1', teremos uma transição deterministicamente para o estado S_1 .

A seguir, apresentamos a definição formal matemática de um autômato finito determinístico:

Seja A um AFD, este terá uma quintupla de variáveis, são elas: $(Q, \Sigma, \delta, q_0, F)$. Estas, por sua vez, podem ser representadas como:

- conjunto finito de símbolos de entrada chamado alfabeto (Σ) ;
- conjunto finito de estados (Q) ;
- função de transição $(\delta : Q \times \Sigma \rightarrow Q)$;
- estado inicial $(q_0 \in Q)$;
- conjunto de estados de aceitação $(F \subseteq Q)$.

Se $w = a_1 a_2 \dots a_n$ é uma cadeia de símbolos do alfabeto Σ , o autômato M terá a cadeia w se, e somente se, existir uma sequência de estados, r_0, r_1, \dots, r_n , em Q , com as seguintes condições:

$$r_0 = q_0,$$

$$r_{i+1} = \delta(r_i, a_{i+1}), \text{ para } i = 0, \dots, n-1$$

$$r_n \in F.$$

Assim, a primeira condição demonstra que a máquina de estados finitos começa a funcionar no estado inicial q_0 .

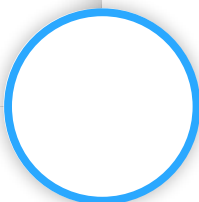
A segunda condição determina que, mediante um dado, teremos cada símbolo da entrada w , logo, a máquina transitará de estado em estado,

segundo a função de transição δ .

Já a terceira condição mostra que a máquina aceitará w se, e somente se, o último símbolo da entrada levar o autômato a parar em um estado f , tal que $f \in F$.

Caso contrário, veremos que a máquina de estados finita irá rejeitar a entrada apresentada. Assim, o conjunto de cadeias que M aceita é chamado linguagem regular por M e é simbolicamente representado por $L(M)$.

SAIBA MAIS



Todo autômato finito determinístico (AFD) possui um estado inicial (denotado graficamente por uma seta de origem anônima) em que sua computação começa e um conjunto de estados finais (denotados graficamente por um círculo de borda dupla), o qual indica a aceitação da cadeia de entrada. Entenda melhor, assistindo à videoaula a seguir:

ASSISTIR

AFDs podem ser empregados para modelar sistemas que validam entradas, como uma mensagem de WhatsApp em um servidor de requisições. Dito isso, vemos que um AFD analisa o conjunto de linguagens regulares que são, dentre outras coisas, fundamentais para a realização de análise léxica e para o reconhecimento de padrões (MENEZES, 2010).

Conhecimento

Teste seus Conhecimentos

(Atividade não pontuada)

Um AFD, ou máquina de estados finita determinística, nada mais é que uma máquina de estados que aceita ou rejeita cadeias de caracteres, gerando um único ramo de computação para cada cadeia de entrada.

DIVERIO, T. A.; MENEZES, P. B. **Teoria da Computação** : máquinas universais e computabilidade. 3. ed. Porto Alegre: Grupo A, 2011. v. 5.

Os AFDs são costumeiramente empregados na modelagem de *softwares* com validação de entradas, tornando, assim, os autômatos finitos determinísticos úteis para a realização de qual tipo de análise?

- ☐ a) Análise sintática.
- ☐ b) Análise lexical.
- ☐ c) Análise de conteúdo.
- ☐ d) Análise gramatical.
- ☐ e) Análise de saídas.

Autômatos Finitos não Determinísticos (AFN)

Depois de termos visto os AFDs, vamos agora trabalhar o conceito dos AFNs. Lembrando que, conforme vimos, um autômato é dito finito por apresentar um conjunto de estados possíveis (Q) finito; no diagrama de estados que apresentamos, por exemplo, temos apenas três estados. Um autômato é determinístico quando, dado o estado atual de M, ao ler um determinado símbolo de entrada, terá apenas um próximo estado possível. Dito isso, vamos tratar dos autômatos finitos não determinísticos (AFN) (MENEZES, 2010).

O AFN ou AFND é uma máquina de estados finita, assim como o autômato finito determinístico, logo, ambos são fundamentais para a teoria da computação. Já estudamos o que é uma máquina de estados finita, no caso dos autômatos finitos não determinísticos, temos que, para cada par de estado e símbolo de entrada, pode haver vários próximos estados possíveis.

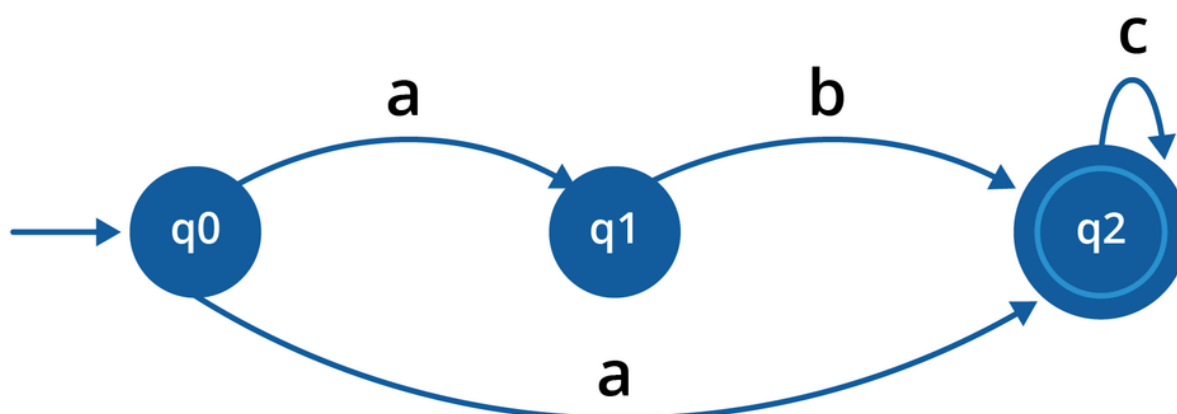
Caro(a) estudante, apesar da distinção do autômato finito determinístico (AFD), em que o próximo estado possível é univocamente determinado, e embora AFD e AFND possuam definições distintas, a teoria formal pode demonstrar que eles são equivalentes; desse modo, para qualquer AFND

dado, pode-se construir um AFD equivalente e vice-versa. Ambos os tipos de autômatos reconhecem apenas linguagens regulares.

Máquinas de estados finitas não determinísticas são generalizadas pelo autômato probabilístico, o qual atribui uma probabilidade para cada transição de estado. Após a criação da teoria dos autômatos, que até 1959 só abarcava os autômatos finitos determinísticos, os professores Michael Rabin e Dana Scott apresentaram a teoria dos autômatos finitos não determinísticos.

O autômato finito não determinístico é uma fundamental generalização presente nos modelos de máquinas, com relevância no estudo dos modelos para concorrência da teoria da computação.

Um AFN, similarmente a um AFD, analisa uma cadeia de caracteres de entrada. Para cada caractere da entrada, há uma transição de um estado para outro, até que todos os caracteres de entrada sejam verificados, porém, existe pelo menos um estado que, ao ler um mesmo caractere, considera mais de uma possibilidade de estado destino. Dessa forma, o próximo estado é um elemento do conjunto das partes dos estados (DIVERIO; MENEZES, 2011). Veja como isso funciona na imagem a seguir:



© VGEducacional

Figura 1.3 - Autômato com duas transições de estados, com transição de estado inicial (q0) e estados subsequentes

Fonte: Diverio e Menezes (2011, p. 89).

#PraCegoVer : na figura apresentada, temos um diagrama de transição de estados, contendo o estado inicial e seus respectivos estados subsequentes, isto é, q1 e q2, com as passagens a e b, respectivamente.

O autômato acima apresenta duas transições de estado como possibilidade ao ler o símbolo “a”, no estado q0. Uma cadeia é aceita por um AFN se, ao serem testadas todas as transições possíveis à medida que se lê a cadeia, o AFN realiza parada em um estado final. Assim, o não determinismo do estado seguinte pode ser interpretado como um teste de todas as possibilidades (DIVERIO; MENEZES, 2011).

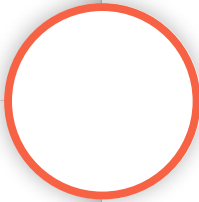
A definição formal matemática de um autômato finito não determinístico é muito similar à do autômato finito determinístico, veja: seja M um AFN, este terá uma quintupla de variáveis: $(Q, \Sigma, \delta, q_0, F)$. Estas, por sua vez, podem ser representadas como:

- um conjunto finito de estados (Q);
- um conjunto finito de símbolos de entrada Σ , o alfabeto;
- uma função de transição ($\delta : Q \times \Sigma \rightarrow P(Q)$);
 - um estado inicial ($q_0 \in Q$);
- um conjunto distinto de estados F como estados de aceitação ($F \subseteq Q$).

Nesse caso, $P(Q)$ é um conjunto das partes de Q . O que muda, então, é a função de transição, que é substituída por uma com cadeia vazia, tendo em vista os diferentes estados possíveis. Desse modo, a função de transição seria:

$$\delta : Q \times \Sigma \cup \{e\} \rightarrow P(Q)$$

REFLITA



“Tanto um sistema de estados finitos determinísticos, como também o não determinístico, opera com estados finitos, ou seja, um sistema de estados finitos é um modelo matemático de sistemas com entradas e saídas discretas. Logo, é possível assumir que, em ambos os cenários, temos um número finito e previamente definido de estados”.

Fonte: Diverio e Menezes (2011, p. 101).

Uma cadeia é rejeitada por um AFN caso haja ausência de caminho de transições que leve o autômato a um estado final após toda a cadeia ser lida. Portanto, podemos concluir que, a cada transição não determinista, temos novos caminhos alternativos possíveis, definindo uma árvore de opções. Uma entrada, por sua vez, é aceita se pelo menos um dos caminhos alternativos recepcionar a entrada (mesmo que os demais não recepcionem). Logo, reflita sobre a ausência de caminho nas transições que levam o autômato a um estado final.

praticar

Vamos praticar

Um autômato é determinístico quando, dado um estado inicial, ao ler um determinado símbolo de entrada, existe apenas um próximo estado possível. Logo, a partir desse raciocínio, podemos afirmar que o autômato finito não determinístico é uma generalização dos modelos de máquinas, sendo de fundamental importância no estudo dos modelos para concorrência, da teoria da computação e das linguagens formais.

Ao analisar um autômato finito não determinístico, vemos que, similarmente a um autômato finito determinístico, ambos vão ler uma cadeia de símbolos de entrada. Em cada símbolo da entrada, há uma transição para um estado novo, até que todos os símbolos de entrada sejam verificados. Logo, qual a diferença fundamental entre o autômato finito determinístico e o autômato finito não determinístico?

Autômatos Finitos com Movimentos Vazios

Neste último tópico, vamos nos aprofundar mais nos autômatos finitos. Os autômatos finitos com movimentos vazios generalizam os movimentos não determinísticos. Dessa forma, o autômato finito com movimentos vazios pode ser interpretado como um não determinismo interno ao autômato, constituindo uma transição encapsulada, ou seja, podemos ter uma exceção por uma eventual mudança de estados (MENEZES, 2010).

1. Uma das vantagens dos autômatos finitos com movimentos vazios, no estudo das linguagens formais, é a praticidade empregada na construção de algoritmos de ordenação.
2. Logo, vale destacar que qualquer autômato finito com movimentos vazios pode ser elaborado a partir de um autômato finito não determinístico.

Por fim, a computação de um autômato finito com movimentos vazios é análoga à de um autômato finito não determinístico. Logo, o processamento de uma transição para uma entrada vazia também é não determinístico. Dessa forma, um autômato com movimentos vazios, ao processar uma entrada vazia, assume simultaneamente os estados destino e origem. Portanto, a origem de um movimento vazio sempre é um caminho alternativo (DIVERIO; MENEZES, 2011).

Agora que temos uma boa base sobre o assunto, nessa jornada que traçamos juntos, podemos praticar um pouco dessa teoria.

praticar

Vamos praticar

A computação de um autômato finito com movimentos vazios é análoga à de um autômato finito não determinístico. Sendo que uma das vantagens dos autômatos finitos com movimentos vazios no estudo das linguagens formais é o fato de facilitarem construções e implementações relacionadas ao desenvolvimento de linguagens de programação orientadas a objetos.

A partir do que foi apresentado, aponte exemplos práticos do seu dia a dia, em que podem ser encontrados autômatos finitos com movimentos vazios. Lembrando que um autômato com movimentos vazios, ao processar uma entrada vazia, assume simultaneamente os estados destino e origem.

Material Complementar



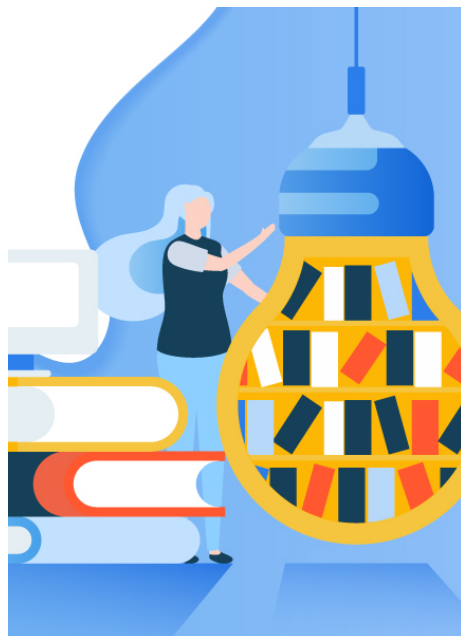
FILME

Pirates of Silicon Valley / “Piratas da Informática”.

Ano: 1999.

Comentário: o filme retrata os bastidores da corrida travada entre Bill Gates e Steve Jobs no desenvolvimento do computador pessoal (PC), onde são demonstrados os fundamentos da elaboração das linguagens de programação e os desafios da época, a fim de conseguir escalar a produção de computadores pessoais e sua respectiva distribuição em todo o mundo. Vale a pena assistir ao filme para compreender como os conceitos apresentados ao longo deste estudo foram fundamentais para criar as duas maiores empresas globais de tecnologia da história.

Para conhecer mais sobre o filme, acesse o *trailer* disponível em:

[ACESSAR](#)

LIVRO

“Estrutura de dados: algoritmos, análise da complexidade e implementações em Java e C/C++”.

Editora: Pearson Prentice Hall.

Autor: Ana F. Gomes Ascencio e Graziela Santos Araújo.

ISBN: 8576058812.

Comentário: o livro trata dos principais conceitos sobre estrutura de dados e algoritmos, demonstrando, na prática, os conceitos de linguagens formais e autômatos, aplicados à linguagem Java e C/C++ na implementação de algoritmos e, respectivamente, nas complexidades dos algoritmos. Vale a pena ler para aprender, na prática, o emprego dos conceitos estudados em conjunto com as linguagens Java e c++.

Caro(a) estudante, chegamos ao fim deste estudo e, após a leitura, podemos concluir que autômato significa "agindo pela vontade própria", conforme demonstramos ao longo da aula. Por sua vez, é um conceito que nasce do próprio estudo da **linguagem natural**, a fim de adaptá-lo para as **linguagens formais**, o que deu origem às **linguagens de programação**. Como a programação tem sido talvez o principal fator de transformação do planeta no último século, é totalmente necessário entender, através dos estudos dos autômatos e das linguagens formais, o surgimento desse paradigma de linguagem computacional que tem evoluído a cada dia.

Referências



ASCENCIO, A. F. G.; ARAÚJO, G. S.

Estruturas de dados : algoritmos, análise da complexidade e implementações em JAVA e C/C++. São Paulo: Pearson Prentice Hall, 2010. v. 3.

DIVERIO, T. A.; MENEZES, P. B. **Teoria da Computação** : máquinas universais e computabilidade. 3. ed. Porto Alegre: Grupo A, 2011. v. 5.

MENEZES, P. B. **Linguagens formais e autômatos** . 6. ed. Porto Alegre: Grupo A,

2010. v. 3.

PIRATAS da Informática. **Adoro Cinema** , 2007. Disponível em:

<https://www.adorocinema.com/filmes/filme-133537/>. Acesso em: 13 maio 2021.

PIRATES of Silicon Valley. Direção/Produção de Martyn Burke. Estados Unidos: TNT, 1999. 1 DVD (95 min.), son., color.

VIDEO Aula - Automato Finito Deterministico. [S. l.: s. n.], 2012. 1 vídeo (24 min 2 s). Publicado pelo canal Wagner Monteiro Azevedo Santos. Disponível em:

<https://www.youtube.com/watch?v=cC8G0xZKqu8>. Acesso em: 13 maio 2021.