Notas de Aula em Linguagens Formais e Autômatos

Maurício Marengoni

Centro Universitário SENAC
mauricio.marengoni@sp.senac.br
http://www.sp.senac.br

1 Autômato, Computabilidade e Complexidade

Os termos citados acima estão ligados na seguinte questão: *O que se pode fazer com um computador e quais são as suas limitações?*

Os problemas resolvidos por computadores aprecem em formatos diferentes, alguns são fáceis e outros são difíceis. A questão envolvida com **Complexidade** é tentar entender o que faz com que um problema seja computacionalmente mais difícil que um outro. A resposta para esta pergunta ainda não é conhecida, porém é possível classificar os problemas de acordo com a sua dificuldade. Uma aplicação direta de complexidade é a área de criptografia.

Computabilidade está relacionado com o fato do que um computador pode resolver ou não. Existem problemas, como por exemplo, determinar se uma sentença matemática é verdadeira ou falsa, que não possuem uma solução algoritmica. Uma das consequencias desta área está na modelagem teórica de computadores que podem auxiliar na construção de computadores reais.

A **Teoria de Autômatos** lida com as definições e propriedades de modelos matemáticos de computação. Existem vários modelos de computação, do mais simples (autômato finito) que pode ser usado no processamento de texto, ao mais complexo (máquinas de Turing) que são modelos de computadores como nós conhecemos hoje.

2 Noções Matemáticas e Terminologia

2.1 Conjuntos

Um conjunto é um grupo de objetos representados como uma unidade. Os objetos de um conjunto são chamados de **elementos** ou **membros** do conjunto. Uma das formas de se representar um conjunto é fazendo uma lista de seus elementos:

$$\{7, 21, Jaqueline, casa, 57\}$$

Os símbolos $\in$ e $\notin$ são utilizados para indicar se um elemento pertence ou não a um conjunto, por exemplo, Jaqueline $\in$ ao conjunto, mas bola $\notin$ ao conjunto. Dados dois conjuntos A e B dizemos que A é um **subconjunto** de B, $A \subseteq B$,

se todos os elementos de A também são elementos de B. A é um subconjunto próprio de B se A for um subconjunto de B mas eles não forem iguais $A \subsetneq B$.

Um **conjunto infinito** contém um número infinito de elementos. Como não é possível listar todos os elementos de um conjunto infinito utilizamos a notação ... para representar uma continuação da sequencia:

$$\{1, 2, 3, \ldots\}$$

Um conjunto especial é o conjunto **vazio** que é o conjunto que não possui elemento algum e é representado por $\emptyset$. Podemos ainda descrever um conjunto listando suas propriedades, por exemplo: $\{p \mid p \text{ é divisível por 2 e } p \geq 0\}$ indica o conjunto de todos os números pares positivos.

Dados dois conjuntos A e B, a **união** de A e B, descrito como $A \cup B$, é o conjunto que contém todos os elementos de A e todos os elementos de B. A **intersecção** de A e B, descrito como $A \cap B$ é o conjunto que contém todos os elementos de A que também são elementos de B. O **complemento** de um conjunto A, descrito como $\overline{A}$ é o conjunto de todos os elementos considerados que não estão em A.

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$
$$A = \{1, 3, 5, 7, 9\} \text{ e } B = \{1, 2, 4, 8\}$$
$$A \cup B = \{1, 2, 3, 4, 5, 7, 8, 9\}$$
$$A \cap B = \{1\}$$
$$\overline{A} = \{2, 4, 6, 8, 10\}$$

2.2 Sequencias e Tuplas

Uma **sequencia** de objetos é uma lista destes objetos em alguma ordem, por exemplo: $(3, 12, 25)$. Num conjunto a ordem não é importante, porém, numa sequencia ela é, logo, $(3, 12, 25)$ é diferente de $(12, 3, 25)$. Seguindo este raciocínio, repetições em uma sequencia fazem diferença, logo, $(3, 12, 25)$ é diferente de $(3, 3, 12, 25)$. Sequencia podem ser finitas ou infinitas, uma sequencia finita é chamada de tupla. Uma sequencia com k elementos é uma k-tupla. No exemplo $(3, 12, 25)$ tem-se um 3-tupla. Uma 2-tupla também é chamado de **par**.

O **conjunto potência** de um conjunto A é definido como o conjunto de todos os subconjuntos de A, logo, se $A = \{a, b\}$ então o conjunto potência de A é dado por $\{\emptyset, \{a\}, \{b\}, \{a, b\}\}$. Se A e B são conjuntos o **produto cartesiano** entre A e B, representado por AxB, é o conjunto de todos os pares (u, v) onde $u \in A$ e $v \in B$.

Exemplo: $A = \{a, b\}$ e $B = \{1, 2, 3\}$ então A x B é dado por:

$$A \text{ x } B = \{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3)\}$$

2.3 Funções e Relações

Uma **função** é um objeto que estabelece uma relação de entrada-saída, isto é, se f é uma função que com entrada igual a a tem como saída o valor b, então podemos escrever que $f(a) = b$.

Uma função também pode ser chamada de um mapeamento, pois, no caso acima, o valor de a é mapeado para o valor de b atarvés da função f. O conjunto de todos os valores possiveis de entrada de uma função é chamado de **domínio** da função e o conjunto de todos os valores possiveis de saída de uma função é chamado de **imagem** ou **contra-domínio** da função. Então podemos escrever $f : D \rightarrow I$ onde D é o dominio de f e I é a imagem de f.

Quando o domínio de uma função f é dado pelo produto cartesiano de vários conjuntos A_1 x A_2 x ... x A_k, a entrada de f é uma k-tupla $(a_1, a_2, \ldots, a_k)$ e os valores a_i são os **argumentos** de f.

Um **predicado** ou **propriedade** é uma função que possui uma imagem igual a $\{VERDADE, FALSO\}$. Uma propriedade onde o domínio é um conjunto de k-tuplas é chamado de **relação**. Uma relação binária é aquela que possui um domínio com pares. Um tipo especial de relação binária é chamada de **relação de equivalencia** e que captura a idéia de dois objetos semelhantes em alguma característica. Uma relação binária R é uma relação equivalente se ela satisfaz as seguintes condições:

- R é reflexiva: ∀ x, xRx.
- R é simétrica: ∀ x e y, xRy → yRx.
- R é transitiva: ∀ x, y e z, xRy e yRz → xRz.

2.4 Grafos

Um **grafo não direcionado** ou simplesmente **grafo** é uma estrutura formada por **nós** ou **vértices** ligados por **arcos** ou **arestas** que indicam algum tipo de relação entre os vértices, conforme indicado na Figura 1

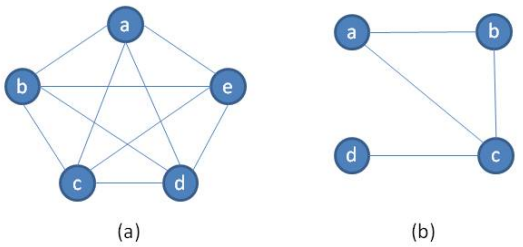

Fig. 1. Dois grafos, em (a) um grafo completo e em (b) simplesmente um grafo

O número de arestas em um determinado vértice é o **grau** do vértice. Num grafo G comos vértices i e j, o par (i, j) representa a aresta que conecta o vértice i com o vértice j. No grafo indireto a ordem não importa. O grafo G é definido pelo conjunto de vértices V e pelo conjunto de arestas E, logo escrevemos $G = (V, E)$. Um grafo pode ser definido por um diagrama ou apresentando os conjuntos V e E.

Dado um grafo $G = (V, E)$, dizemos que $H = (V_H, E_H)$ é um **subgrafo** de G se $V_H \subseteq V$ e $E_H \subseteq E$.

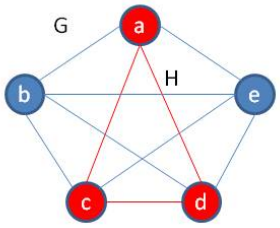

Fig. 2. Em vermelho um subgrafo H do grafo G.

Um **caminho** em um grafo é uma sequencia de vértices conectados por arestas. Um **caminho simples** é uma caminho onde os vértices não aparecem repetidos. Um grafo é chamado de **conectado** se existir um caminho que conecta qualquer dois vértices do grafo. Um caminho é chamado de **ciclo** se ele começa e termina no mesmo vértice. Um **ciclo simples** contém pelo menos 3 vértices e repete apenas o primeiro e o último vértice. Uma **árvore** é um grafo conectado que não possui ciclos. Uma árvore possui um vértice especial que é chamado de **raiz**. O vértices de grau 1 de uma árvore que não é a raiz, são chamados de **folhas**.

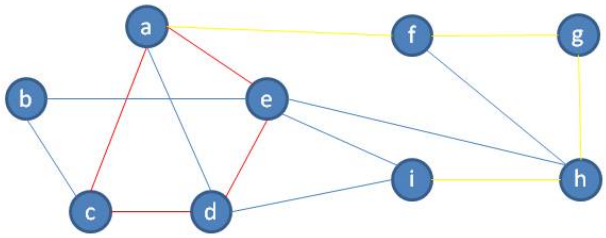

Fig. 3. Em vermelho um ciclo no grafo e em amarelo um caminho ligando o vértice a ao vértice i.

Um **grafo direcionado** é um grafo onde as arestas possuem uma direção, ou que a relação entre os vértices é unilateral, neste caso, uma aresta (i, j) indica que ela vai do vértice i para o vértice j. O número de arestas que saem de um vértice num grafo direcionado é chamado de **grau de saída** do vértice e o número de arestas que chegam em um vértice num grafo direcionado é chamado de **grau de chegada** do vértice.

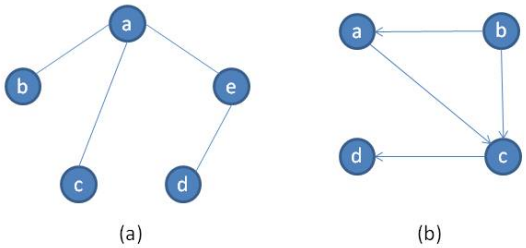

Fig. 4. Em (a) temos um exemplo de uma árvore e em (b) temos um exemplo de grafo direcionado.

Um **caminho direcionado** é um caminho onde a sequencia de vértices representa arestas existentes no grafo. Um grafo direcionado é chamado de **fortemente conectado** se existir um caminho direcionado conectando qualquer par de vértices. Os grafos direcionados são uma forma gráfica de representação de uma relação binária.

2.5 Palavras e Linguagens

Cadeias de caractéres são blocos construtivos importantes em ciência da computação. Um **alfabeto** é um conjunto finito e não vazio de **símbolos**. Os alfabetos são representados por letras gregas maiusculas (Σ, Γ). Uma **palavra** é uma sequencia finita de símbolos de um alfabeto. Se w é uma palavra no alfabeto Σ, o **comprimento** de w, escrito como $|w|$ é o número de símbolos em w. Uma palavra com comprimento zero é chamada de **palavra vazia** e é representada pelo símbolo ϵ.

$$\Sigma = \{0, 1, a, b, c\}$$
w = 001bbc00 é uma palavra em Σ de comprimento = 8, mas v = 00ABC11 não é uma palavra em Σ

Se w é uma palavra do alfabeto Σ e possui um comprimento n podemos então escreve-la como uma sequencia de n símbolos: $w = w_1 w_2 w_3 \dots w_n$ onde cada w_i é um símbolo do alfabeto Σ. O reverso de w é denominado $w^R = w_n w_{n-1} \dots w_1$. Uma palavra v é uma sub-palavra de w se v aparece numa sequencia dentro de w, exemplo: $w = 0010100101$ e $v = 1010$. A **concatenação** de duas palavras w e v é uma palavra wv com todos os símbolos de w seguidos pelos símbolos de v.

Uma **ordem lexicográfica** de palavras são palavras escritas em ordem, como se fosse um dicionário. Dado um alfabeto $\Sigma = a, b$, as palavras sobre este alfabeto em ordem lexicográfica são: $\{\emptyset, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$

Uma **linguagem** é um conjunto de palavras que possuem alguma propriedade comum.

2.6 Lógica Booleana

A **Lógica Booleana** é um sistema matemático baseado em dois valores $FALSO$ e $VERDADEIRO$ que são chamados de **valores Booleanos** e que são muitas vezes substituídos pelos valores 0 e 1. A lógica Booleana possui um connjunto de operações, a mais simples delas é a **NEGAÇÂO** ou **NÃO**, definida pelo símbolo $\neg$ e que inverte o valor, por exemplo: $\neg 0 = 1$. A **CONJUNÇÃO** ou **E** é representada pelo símbolo $\wedge$ é uma operação sobre dois valores Booleanos que retorna o valor 1 se e somente se os dois valores Booleanos forem iguais a 1. A **DISJUNÇÃO** ou **OU** é representada pelo símbolo $\vee$ e também é uma operação sobre dois valores Booleanos que retorna 1 se pelo menos um dos valores Booleanos for igual a 1. A tabela abaixo apresenta um resumo dos operadores Booleanos descritos e também o resultado dos operadores **OU EXCLUSIVO**, **IMPLICA** e **SE E SOMENTE SE**.

A	B	$\neg A$	A $\wedge$ B	A $\vee$ B	A $\oplus$ B	A $\rightarrow$ B	A $\leftrightarrow$ B
0	0	1	0	0	0	1	1
0	1	1	0	1	1	1	0
1	0	0	0	1	1	0	0
1	1	0	1	1	0	1	1

A operação distributiva dos operadores E e OU é definida da seguinte forma:

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$$
$$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$$

3 Definições, Teoremas e Provas

Uma **definição** descreve matematicamente um objeto, a precisão é essencial na definição matemática do objeto que deve incluir tudo que faz parte do objeto e excluir o que não faz parte do objeto. **Afirmações matemáticas** são expressões que mostram as propriedades de um certo objeto. Uma afirmação pode ou não ser verdadeira, porém, ela deve ser precisa e não permitir ambiguidades.

Uma **prova** é uma sequencia lógica de argumentos que demonstra que uma afirmação matemática é verdadeira. Um **teorema** é uma afirmação matemática que foi provada como verdadeira, porém, este termo é utilizado apenas para afirmações especiais. Quando a prova de uma afirmação apenas serve como base para a prova de uma outra afirmação mais interessante, nestes casos usamos o termo **lema**. Ainda, quando a prova de um teorema leva a conclusões sobre a veracidade de outras afirmações, estas outras afirmações são chamadas de **corolários**.

3.1 Provas

A única forma de se provar a veracidade ou não de uma afirmação matemática é através de uma prova matemática, porém, encontrar estas provas não é uma tarefa fácil. Embora não exista uma receita de como fazer uma prova matemática, algumas estratégias podem ser seguidas:

- Leia a afirmação e tenha certeza de ter entendido a afirmação. Se necessário quebre a afirmação em partes e considere as partes em separado.
- Para cada afirmação a ser provada tente verificar intuitivamente sobre a veracidade da afirmação. Se um objeto possui uma certa propriedade veririfique alguns exemplos deste objeto para ver se eles possuem a propriedade ou se existe um **contra-exemplo**, isto é, um objeto que não possui esta propriedade.
- Caso esteja com dificuldade ainda em encontrar a prova, verifique alguns casos especiais, por exemplo, se algo deve ser verdadeiro para todo valor positivo, tente mostrar que isto é verdade quando o valor é igual a 1, e depois quando o valor é igual a 2 até que voce tenha um entendimento maior da afirmação.
- Uma vez encontrada a prova escreva a prova com cuidado, passo a passo, onde cada passo deve ser compreendido facilmente a partir do passo anterior.

Uma prova, geralmente vem na forma $A \Leftrightarrow B$ que quer dizer que A é verdadeiro se e somente se B for verdadeiro onde A e B são afirmações matemáticas. Note que neste caso a prova pode ser quebrada em duas partes. A primeira parte A somente se B, que quer dizer que se A é verdadeiro então B também é $(A \Rightarrow B)$ é o que chamamos de **prova para frente**. Na segunda parte A se B, que quer dizer que se B é verdadeiro então A é verdadeiro $(A \Leftarrow B)$ é o que chamamos de **prova reversa**. Muitas provas matemáticas podem ser decompostas desta forma.

Algumas dicas para executar uma prova:

- Seja paciente: dificilmente uma prova matemática será feita rapidamente e na primeira vez que for tentada, provas levam tempo para serem construidas.
- Não desista: olhe a prova, quebre em partes, analise cada parte separadamente. Se não encontrar a solução, deixe de lado e volte mais tarde, as vezes é necessário algum tempo para amadurecer as idéias.
- Seja claro: ao construir uma intuição ou prova sobre uma afirmação matemática, use esquemas ou diagramas ou mesmo textos que sejam simples e claros.
- Seja objetivo: expresse suas idéias de forma objetiva, utilize notação matemática adequada e demonstre claramente a sua linha de raciocínio.

Exemplo: Para qualquer dois conjuntos A e B, $\overline{A \cup B} = \overline{A} \cap \overline{B}$.

Note que esta é uma prova do tipo descrito acima e que precisa ser provada nas duas direções. Na prova para frente basta seguir os passos:

1. Seja p um elemento que está em $\overline{A \cup B}$, logo, p não pode estar em $A \cup B$ pela definição de complemento.
2. Se p não está em $A \cup B$ então p não está em A e p não está em B, pela definição de união.
3. Isto implica que p tem que estar em $\overline{A}$ pois $p \notin A$ e, analogamente, p tem que estar em $\overline{B}$ pois $p \notin B$
4. Como $p \in \overline{A}$ e $p \in \overline{B}$ pela definição de intersecção $p \in \overline{A} \cap \overline{B}$, como queriamos provar.

Tente fazer a prova no sentido reverso.

4 Tipos de Provas

Existem várias técnicas de provas que podem ser usadas, porém, geralmente, uma determinada técnica de prova é mais adequado para um certo tipo de prova.

4.1 Construção

Na técnica de prova por construção a idéia é de se mostrar como construir um objeto que possui uma determinada propriedade.

4.2 Contradição

Na técnica de prova por contradição a idéia é assumir que o teorema a ser provado é falso e então mostrar que a consequencia da falsidade leva a uma contradição.

4.3 Indução

A técnica de prova por indução é utilizada para provar que todos os elementos de um conjunto possuem uma certa propriedade. Está técnica é considerada uma técnica avançada de prova e é dividida em duas partes: prova-se o caso base; assume-se que o teorema é verdadeiro num caso geral e prova-se a etapa indutiva.

References

1. Sipser, M.: Introdução à Teoria da Computação, Thomson, 2a edição americana, 2007.
2. Sipser, M.: Introduction to the Theory of Computation, Thomson, 2th edition, 2006.
3. Menezes, P.B.: Linguagens Formais e Autômatos, Série Livros Didáticos Instituto de Informática de UFRGS, Sagra Luzzatto 2005.
4. Vieira, N.J.: Introdução aos Fundamentos da Computação, Linguagens e Máquinas, Scott, Thomson, 2006.
5. Solow, D.: How to Read and Do Proofs, an introduction to mathematical thought processes, John Wiley and Sons, 2nd edition, 1990.