Conceitos Básicos

Vocabulário

Cadeias

Linguagens

Problema

Alfabeto ou Vocabulário: Conjunto finito não vazio de símbolos. Símbolo é um elemento qualquer de um alfabeto.

Ex: {A,B,C,….Z} alfabeto latino (maiúsculas)

{α, β, γ, δ, ε, …, ω} alfabeto grego

{0,1} alfabeto binário

{0,1,2,3,4,5,6,7,8,9} alfabeto de dígitos

{a, b}

Cadeia ou palavra: Concatenação de símbolos de um alfabeto. Define-se como cadeia vazia ou nula uma cadeia que não contém nenhum símbolo.

Ex: aab

123094

λ – cadeia nula

Comprimento de cadeia: Número de símbolos de uma cadeia.

Ex: $|aab| = 3$

$|123094|=6$

$|\lambda| = 0$

Concatenação de cadeias: Define-se a concatenação z de uma cadeia x com uma cadeia y como sendo a concatenação dos símbolos de ambas as cadeias, formando a cadeia xy.

$|z| = |x| + |y|$

Ex:

$x = abaa;\ y = ba \Rightarrow z = abaaba\ |z|=4+2=6$

$x = ba;\ y = \lambda \Rightarrow z = ba\ |z|=2+0=2$

Produto de alfabetos: É o produto cartesiano de alfabetos.

Ex:

$V1 = \{a,b\}$; $V2 = \{1, 2, 3\} \Rightarrow V1.V2 = V1xV2 =$
$\{a1, a2, a3, b1, b2, b3\}$

Observe que $V1.V2 \neq V2.V1$

Exponenciação de alfabetos: São todas as cadeias de comprimento n sobre V (V^n). $V^0=\{\lambda\}$, $V^1=V$, $V^n=V^{n-1}.V$

Ex:

$V = \{0, 1\}$

$V^3 = V^2.V = (V.V).V = \{00, 01, 10, 11\}.\{0,1\} = \{000, 001, 010, 011, 100, 101, 110, 111\}$ $|V|=8$

$|V^n| = m^n$ onde $|V| = m$

Fechamento (Clausura) de um Alfabeto:

Seja A um alfabeto, então o fechamento de A é definido como

$$A^* = A^0 \cup A^1 \cup A^2 \cup \ldots \cup A^n \cup \ldots$$

Portanto A^* = conjunto das cadeias de qualquer comprimento sobre o alfabeto A.

Ex:

$A = \{1\}$

$A^* = \{\lambda, 1, 11, 111, \ldots\}$

Fechamento Positivo de A: $A^+ = A^* - \{\lambda\}$ (todas as cadeias não vazias sobre o alfabeto A)

Linguagem é um conjunto de *cadeias* de símbolos sobre um alfabeto/vocabulário, V. É um subconjunto específico de V*. Estas cadeias são denominadas *sentenças* da linguagem, e são formadas pela justaposição de elementos individuais, os símbolos da linguagem.

Ex: $V = \{a, b, c\}$

$L = \{ab, bc\}$ (linguagem formada pelas cadeias ab e bc)

Formadores de Conjuntos

Expressões que ajudam a descrever uma linguagem (em geral infinita):

$L = \{ w \mid \text{algo sobre } w \}$

O conjunto de palavras w tais que (o que for dito sobre w à direita)

Exs.:

1. L={ w | w consiste de um número igual de 0's e 1's }
2. L={ w | w é um número inteiro binário primo }
3. L={ w | w é um programa em C sintaticamente correto }
4. $L=\{ 0^n 1^n \mid n \geq 1\} = \{01, 0011, 000111, ...\}$
5. $L=\{0^i 1^j \mid 0 \leq i \leq j\} = \{\lambda, 01, 011, 011, ..., 0011, 00111, ...\}$
6. $L=\{ab^n \cup a^nb \mid n>=0\}$ (linguagem formada por todas as cadeias que começam com "a" seguido de um número qualquer de "b"'s OU começam com um número qualquer de "a"'s seguidos de um "b", por exemplo a, b, ab, abb, aab, aaab, ...)

Um **Problema** na Teoria dos Autômatos consiste em decidir se uma dada cadeia é elemento (sentença) de alguma linguagem específica. Assim, um problema é sinônimo de decidir a pertinência a um conjunto (uma linguagem). Mais precisamente:

Se Σ é um alfabeto e L é uma linguagem sobre Σ, então o **problema L** é:

Dada uma cadeia w em Σ^*, decidir se w está ou não em L. (Questão de Decisão)

Ex.: O problema de testar o caráter primo de um número binário pode ser expresso pela linguagem Lp que consiste em todas as cadeias binárias cujo valor como número binário é primo. Ou seja, dada uma cadeia de 0's e 1's, diremos *sim* se a cadeia for a representação binária de um primo e diremos *não* caso contrário. Lp é então o conjunto-solução do problema.

Linguagem Lp ou Problema P?

- São dois lados da mesma moeda.
- A Linguagem Lp é o <u>conjunto das soluções </u>de um problema P.
- Todo problema P tem um problema de decisão associado: dada uma (cadeia) candidata à solução, decidir se ela de fato é solução (se tem as características de solução). Ou seja, se pertence à Lp.
- Diferentemente, um algoritmo que **busca** uma solução tem o trabalho extra de verificar mais de uma possibilidade.

Encarar problemas como uma questão de pertinência de conjunto (pertence ou não à linguagem) tem sido útil aos estudos da teoria da complexidade.

Problemas de Decisão têm se mostrado tão difíceis quanto suas versões do tipo "resolva isso".

Ao reduzirmos um problema para sua versão de pertinência e provarmos que é difícil resolvê-la, então podemos concluir que resolver o problema inicial será igualmente difícil. **(Técnica de Prova por Redução)**

$$\boxed{P \equiv i \in Lp\ ?}$$

Exemplo

- Seja o problema P1, de compilar um programa na linguagem de programação X. Quão difícil é resolvê-lo? (qual sua complexidade?).
- Considere agora o problema de decisão P2: dada uma cadeia, ela pertence à linguagem Lx (de cadeias válidas na linguagem de programação X)?
- Se provarmos que é difícil resolver P2 (H), então **não será mais fácil** converter programas na linguagem X para código-objeto (C).
- Por contrapositiva: se fosse fácil (eficiente) gerar código-objeto, poderíamos usar o próprio compilador para, ao ter sucesso ao produzir o código para um programa (cadeia), concluir que essa entrada se trata de um elemento válido de Lx. Assim, seria fácil testar a pertinência. Ou seja, se $\sim C \rightarrow \sim H$.

Assim, temos uma prova por contrapositiva da afirmação:

Se o teste de pertinência a Lx é difícil, então compilar programas na linguagem de programação X é difícil.

Essa técnica de redução é extremamente útil no estudo da complexidade de problemas, e é facilitada pela noção de que problemas são questões sobre pertinência a uma linguagem, e não tipos mais gerais de questões.