1 Introdução

1.1 Motivação

Qualquer que seja a área cientifica de estudo, existem limites que não podem ser superados, qualquer que seja a tecnologia utilizada.

Existem problemas da matemática que não podem ser resolvidos por um programa de computador.

Tais problemas devem ser identificados.

Para isso, precisamos formalizar as noções de:

- Problema (Como dar uma representação genérica de qualquer problema? Numa forma que o computador vai entender?)

- Programa (O que é exatamente um algoritmo em computação ?)

1.2 Noção de Problema

Ex: Determinar se um número natural é par é um problema.

Caraterísticas de um problema:

- um problema é uma questão genérica que se aplica a um conjunto de elementos (ex: os naturais)

- cada instancia de um problema (uma questão que corresponde a um elemento particular do conjunto estudado) tem uma resposta : 35 é par? Resposta: N

- a noção de problema é independente da noção de programa : não é o programa que define o problema (existem vários programas para resolver um mesmo problema)

- para resolver um problema utilizando um programa é necessário fixar uma representação das instancias do problema (é sobre tal representação que o programa atuará)

Exemplos de problemas:

- ordenar um vetor de números é um tipo de problema que um programa pode resolver

- determinar se um programa escrito em C para qualquer que sejam os dados de entrada (problema da parada) : detectar a existência de loop ∞

Quando a resposta de um problema é sempre SIM ou NÃO o problema é chamado de binário (o problema da parada é um problema binário)

Pode-se verificar que os resultados obtidos para a classe dos problemas binários podem ser entendidos aos problemas não binários (problema de otimização num grafo por exemplo)

1.3 Noção de Programa

Um programa (escrito em C por exemplo) que sempre fornece uma resposta para um problema dado é chamado de Procedimento Efetivo.

Uma linguagem de programação ficará associada à noção de procedimento efetivo somente se existe um procedimento de interpretação ou compilação que transformará o programa num código diretamente executado pelo processador.

Para formalizar a noção de procedimento efetivo, serão utilizados formas de linguagens de programação tão simples que o procedimento de interpretação será imediato : tais programas serão chamados de Autômatos.

E a noção de Algoritmo ?

Até hoje não existe uma definição única e aceita por todos.

Para Minsky (1967) por exemplo, um algoritmo (um conjunto de regras que especificam a cada instante o próximo comportamento) é sinônimo de procedimento efetivo.

Um dos objetivos da disciplina é de formalizar a noção de procedimento efetivo e de analisar os procedimentos efetivos utilizados como solução de problemas.

1.4 Formalização da noção de Problema

Para poder estudar a noção de procedimento efetivo, independentemente do programa e da máquina que o executa, a representação dos dados de entrada do problema deve fazer abstração dos tipos de dados particulares utilizados pelas linguagens de programação.

Cada instância do problema será então representada por uma seqüência finita de símbolos que pertencem a um conjunto de símbolos que dependem do problema.

Exemplos :

- {0,...,9} para um problema sobre os inteiros

- {a,...,z} para um problema sobre as instruções de uma linguagem de programação

1.4.1 Noção de Alfabeto e Palavras

Def: Um alfabeto Σ é um conjunto finito de símbolos.

Def: Uma palavra w (cadeia, string, seqüência) definida sobre um alfabeto é uma seqüência finita de elementos do alfabeto.

O comprimento de uma palavra w será anotado |w|

|w| pode ser igual a 0 : trata-se então da palavra vazia anotada geralmente e, ε ou λ

1.4.2 Representação dos Problemas

Todos os dados manipulados em computação podem ser representados por cadeias de caracteres

Ex: uma imagem será representada por uma seqüência das intensidade dos seus pontos.

Seja um problema binário cujas instancias foram codificadas por palavras definidas num alfabeto Σ.

O conjunto de todas as palavras definidas em Σ pode ser particionado em 3 subconjuntos:

- as palavras que representam as instâncias do problema tais que a resposta do problema é SIM (instâncias positivas do problema)
- as palavras que representam as instâncias do problema tais que a resposta do problema é NÃO (instâncias negativas do problema)
- as palavras que não representam instâncias do problema

Ex: alfabeto $\Sigma = \{0,...,9,a,...,z\}$

Problema dos números pares

19 resposta NÃO

26 resposta SIM

2a9 resposta : não é uma instância do problema (é um tipo de NÃO)

As duas últimas classes podem ser consideradas como uma única classe.

Conclusão: Um problema pode ser caracterizado pelo conjunto das palavras que representam as instâncias positivas do problema.

1.4.3 Noção de Linguagem

Def: Uma linguagem é um conjunto de palavras definidas a partir de um mesmo alfabeto.

Um problema é caracterizado pela linguagem que codifica as instâncias positivas do problema.

A resolução de um problema consiste então em reconhecer as palavras de uma linguagem que caracteriza as instâncias positivas de um problema.

Ex:

{ε,aaaaa,a,bbbbbb} e {aab,aaaa,ε,a,b,abb} são linguagens (finitas) no alfabeto {a,b}

{ε,0,1,00,01,10,11,000,001,...} é a linguagem de todas as palavras que podem ser formadas no alfabeto {0,1}

A linguagem Ø representa a linguagem vazia que não contém nenhuma palavra. É uma linguagem diferente da linguagem {ε} (linguagem que contém a palavra vazia).

O conjunto as palavras utilizadas para escrever programas em C é uma linguagem.

1.5 Representação das linguagens

Não existe uma representação (notação) que descreve qualquer linguagem!

Se a linguagem é finita, a enumeração explícita das palavras que pertencem à linguagem é suficiente para definir a linguagem.

Se a linguagem é infinita a enumeração explícita não corresponde a uma possibilidade de descrição da linguagem.

1.4.1 Operação sobre as linguagens

Sejam 2 linguagens L1 e L2.

A união de L1 e L2 é uma linguagem que contém todas as palavras contidas tanto em L1 quanto em L2.

$L1 \cup L2 = \{w/w \in L1 \text{ ou } w \in L2\}$

A concatenação de 2 linguagens L1 e L2 é a linguagem que contém todas as palavras formadas de uma palavra de L1 seguida de uma palavra de L2

$L1 \,.\, L2 = \{w/w = xy \text{ com } x \in L1 \text{ e } y \in L2\}$

O fechamento iterativo de uma linguagem L1 ou fechamento de KLEENE é o conjunto das palavras formadas pela concatenação finita de palavras de L1

$L1^* = \{w/ \exists k \geq 0$ e $w1,w2,...,wk \in L1$ tais que $w=w1w2w3...wk\}$

O complemento de uma linguagem é o conjunto de todas as palavras que não pertencem a aquela linguagem

$\{w/w \notin L1\}$ (palavras formadas no mesmo
beto que L1)

Def: O conjunto $\mathcal{R}$ das linguagens regulares num alfabeto Σ é o menor conjunto que satisfaz as seguintes condições :

(1) $\emptyset \in \mathcal{R}$; $\{\varepsilon\} \in \mathcal{R}$;

(2) $\{a\} \in \mathcal{R}$ para todo $a \in \Sigma$;

(3) se A e B $\in \mathcal{R}$ então $A \cup B$, A.B e $A^* \in \mathcal{R}$

O menor conjunto significa que somente os conjuntos construídos a partir dos conjuntos elementares $\emptyset$, $\{\varepsilon\}$ e $\{a\}$ para as operações sobre as linguagens pertencem a $\mathcal{R}$

Ex:

Se $L = \{001, 10, 111\}$

e $M = \{\varepsilon, 001\}$

ENTÃO

$L \cup M = \{\varepsilon, 10, 001, 111\}$

$L \cdot M = \{001, 10, 111, 001001,$
$10001, 111001\}$

SE $L = \{0, 11\}$ ENTÃO

$L^* = \{011, 11110, \varepsilon, \ldots\}$

01011 ou $101 \notin L^*$

1.4.2 Expressões regulares

Uma expressão regular é uma notação utilizada para representar uma linguagem regular.

Tal notação indica como um conjunto regular é obtido a partir dos conjuntos regulares elementares.

Def: As expressões regulares para um alfabeto Σ são as expressões formadas pelas seguintes regras:

(1) Ø, ε e os elementos de Σ são expressões regulares

(2) se α e β são expressões regulares então

$(\alpha \cup \beta)$, $(\alpha\beta)$, $(\alpha)^*$ são também expressões regulares

Uma expressão regular representa uma linguagem regular.

Def: A linguagem $L(\xi)$ representada pela expressão regular ξ é definida da seguinte forma:

(1) $L(\emptyset) = \emptyset$, $L(\varepsilon) = \{\varepsilon\}$

(2) $L(a) = \{a\}$ para todo $a \in \Sigma$

(3) $L((\alpha \cup \beta)) = L(\alpha) \cup L(\beta)$

(4) $L((\alpha\beta)) = L(\alpha) . L(\beta)$

(5) $L((\alpha) \quad) = L(\alpha)^*$ (a linguagem representada pela expressão regular $(\alpha) \quad$ é o fechamento de KLEENE das palavras da linguagem $L(\alpha)$

Teorema: Uma linguagem é regular se e somente se ela pode ser representada por uma expressão regular.

Ex:

1) Representar o conjunto de todas as palavras obtidas a partir de $\Sigma = \{a1,a2,...,an\}$.

2)Representar o conjunto de todas as palavras diferentes de ε e definidas a partir de $\Sigma = \{a1,a2,...,an\}$.

3) Qual é a linguagem representada pela expressão regular $(a \cup b)^* \; a(a \cup b)^*$?

4) Escrever uma expressão regular para o conjunto de string que consistem em 0's e 1's alternados.

1.5 Linguagens não regulares

Existem linguagens não regulares : que não podem ser definidos através de uma expressão regular.

Isso vem da seguinte observação: não existem expressões regulares suficientes para representar todas as linguagens.

Isso vem do fato que todos os conjuntos infinitos não tem a mesma cardinalidade (mesmo tamanho) **!**

A cardinalidade de um conjunto finito corresponde ao número de elementos que constituem o conjunto.

2 conjuntos têm mesmo tamanho (cardinalidade) se podemos associar cada elemento do primeiro com o elemento do segundo.

Formalmente, tem de existir uma função bijetora entre os dois conjuntos.

A função f: E→F é bijetora se ∀y de F, a equação em x: f(x)=y tem em E uma única solução.

Ex: os conjuntos {0,1,2,3} e {a,b,c,d} têm o mesmo tamanho. Existe por exemplo uma função bijetora que fornece como resultado as seguintes associações: {(0,a),(1,b),(2,c),(3,d)}

Isso significa que a cardinalidade dos conjuntos é a mesma.

Existe também a noção de cardinalidade de um conjunto infinito.

O primeiro e menor conjunto infinito é o conjunto dos números naturais N={0,1,2,...}.

Usando a noção de função bijetora, podemos identificar conjuntos infinitos de mesma cardinalidade que o conjunto N.

Def: um conjunto infinito é enumerável se existe uma função bijetora entre o conjunto estudado e o conjunto dos naturais.

Ex: o conjunto dos números pares é enumerável. A função bijetora de tal conjunto com o conjunto do naturais N produz:

{(0,0),(2,1),(4,2),(6,3),...}

Ex: o conjunto das palavras formadas a partir do alfabeto {a,b} é enumerável. Para provar tal afirmação, podemos classificar as palavras por ordem crescente respeitando a ordem lexicográfica no caso de palavras de mesmo tamanho.

{(ε,o),(a,1),(b,2),(aa,3),(ab,4),(ba,5),(bb,6),
(aaa,7),...}

O conjunto dos racionais é enumerável **!**
Prova: os racionais são caracterizados por um par de números naturais a/b tal que b≠0 e tal que não existem fatores comuns entre a e b. Podemos então classificar pares de naturais que satisfazem tal restrição seguindo a ordem crescente da soma a+b e considerando para cada valor igual desta soma a ordem de numerador crescente. Obtemos então como resultado da função bijetora:

{(0/1,0),(1/1,1),(1/2,2),(2/1,3),(1/3,4),(1/3,4),
(3/1,5),...}

$\frac{0}{1} \Rightarrow a+b=1$

$\cancel{\frac{1}{0}} \Rightarrow$ PROIBIDO

$\frac{0}{2} \Rightarrow a+b=2$ (JÁ EXISTE)

$\frac{1}{1} \Rightarrow a+b=2$

$\frac{0}{3} \Rightarrow a+b=3$ (JÁ EXISTE)

$\frac{1}{2} \Rightarrow a+b=3$

$\frac{2}{1} \Rightarrow a+b=3$

As expressões regulares são enumeráveis **!**
Prova: as expressões regulares são cadeias de carácteres num alfabeto finito. O conjunto de todas as cadeias num alfabeto é enumerável (ver exemplo Σ={a,b}). As expressões regulares correspondem a um sub-conjunto infinito das cadeias de carácteres formadas pelo alfabeto Σ e são então enumeráveis (ver exemplo dos números pares que correspondem a um sub-conjunto dos naturais).

Parece que todos os conjuntos infinitos são enumeráveis. Isso não é verdade **!**

Exercício: Provar que o produto cartesiano $X \times Y = \{(x,y) / x \in X \text{ e } y \in Y\}$ de dois conjuntos enumeráveis é também enumerável.

Teorema: o conjunto dos subconjuntos de um conjunto infinito enumerável não é enumerável !

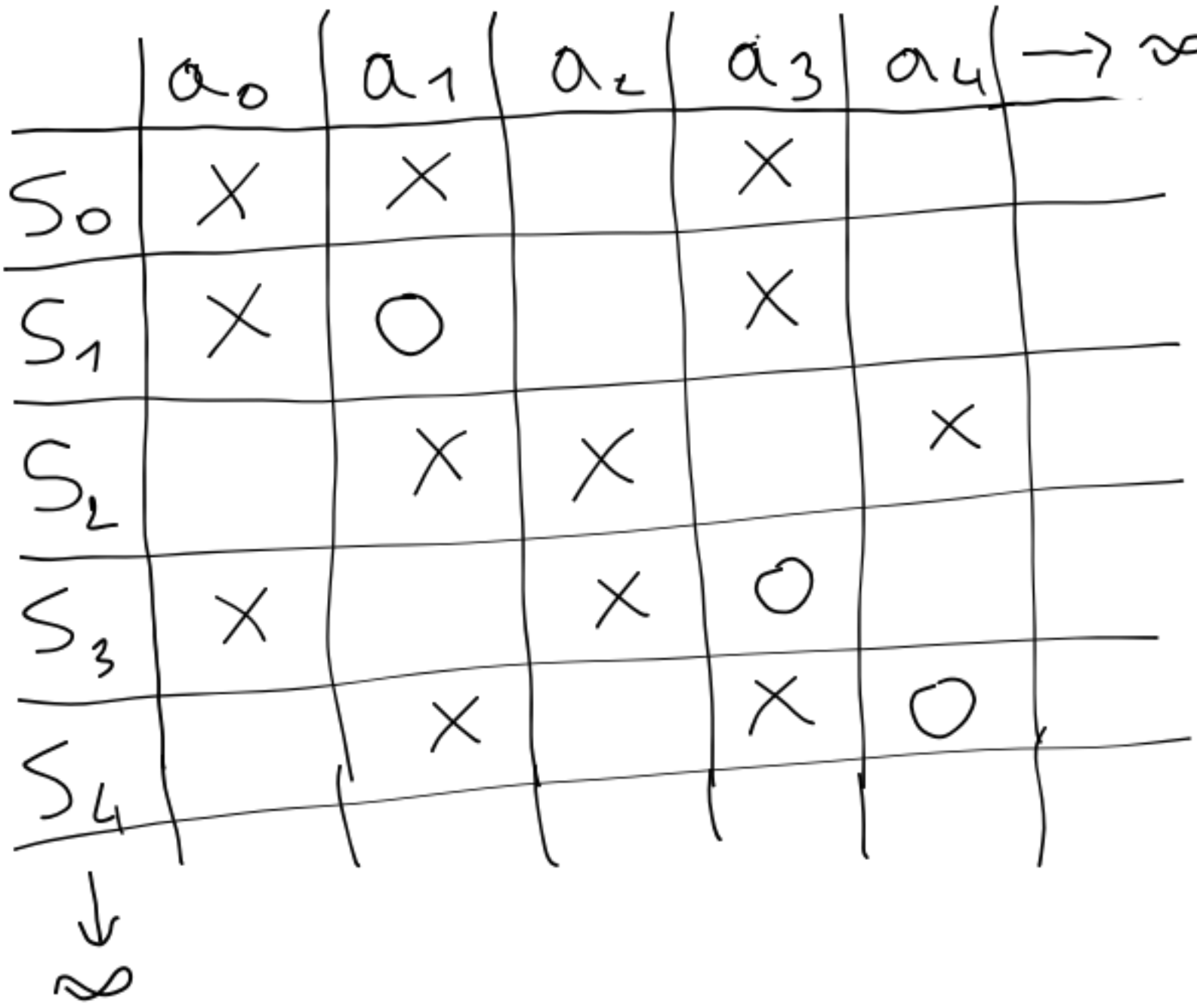
Prova por contradição : em lógica das proposições, em vez de provar S ⇒ P com S e P proposições, podemos provar de modo equivalente

S e não P ⇒ Falso (contradição)

A prova deste teorema é baseado na técnica da diagonalização: se fosse possível enumerar o conjunto S dos sub-conjunto formados por elementos do conjunto enumerável A={a0,a1,a2,...}, seria então possível de produzir um conjunto S={S0,S1,S2,...} para definir a função bijetora {(a0,S0),(a1,S1),...}

Em lógica, queremos provar então que A⇒não S ou A e S ⇒ Falso (contradição)

Se S existe, então podemos produzir a seguinte tabela infinita:



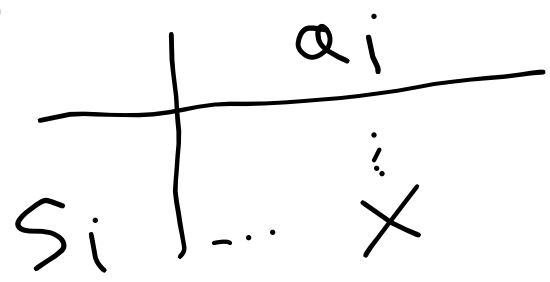
Um "x" indique que um elemento ai ∈ Sj (por exemplo, a2∈S3)

Podemos considerar o conjunto seguinte:

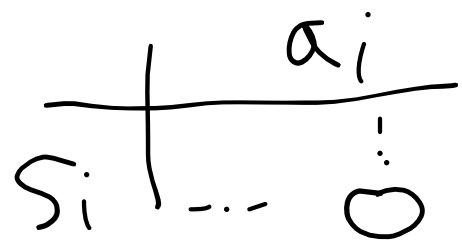
$$D=\{ai / ai \notin Si\}$$

Tal conjunto é representado pelo símbolo "o" na tabela infinita. Tal conjunto de elementos corresponde bem a um sub-conjunto de elemento de A. O conjunto D deveria então ser um elemento Si de S ! Se D=Si existe então um elemento ai de A que então pertence a D(e Si) ou que não pertence a D(e Si)

Se ai $\in$ Si então ai $\notin$ D pela própria definição do conjunto D



Se ai $\notin$ Si então ai $\in$ D pelo própria definição do conjunto D



Encontramos então uma contradição !

Podemos concluir que D não pode ser representado por nenhum elemento Si de S, o que corresponde à uma contradição já que D é um elemento de S.

Isso prova que na verdade, o conjunto S não é enumerável e tem uma cardinalidade infinita maior que o conjunto N !

É sobre as linguagens regulares ?

Considerando um alfabeto Σ, o conjunto das linguagens é o conjunto dos subconjuntos de um conjunto enumerável (o conjunto de todas as palavras que podem ser formadas pelo alfabeto Σ).

O conjunto das linguagens que podem ser formadas a partir de Σ não é então enumerável.

O conjunto das linguagens regulares é enumerável já que cada linguagem regular é representado por uma expressão regular e que o conjunto das expressões regulares é enumerável.

Conclusão : existem muito mais linguagens que as linguagem regulares !

Como tal resultado tem uma influência sobre a decidibilidade ou a não decidibilidade dos problemas em Computação ❓

O que acontece é que um problema será tratado por um programa (uma máquina de Turing) em Computação. E o conjunto infinito de programa que se pode escrever numa linguagem de programação é enumerável (provar tal afirmação). Mas o conjunto infinito de problemas (linguagens) em matemática não é enumerável ❗ Não existirá então uma função bijetora para associar cada problema a um programa. Existe então mais problema que programas ❗