# 7 Linguagens Recursivamente Enumeráveis

Uma linguagem $L$ é dita **recursivamente enumerável** (ou simplesmente **irrestrita**) se for aceita por pelo menos uma Máquina de Turing $M$. Ou seja:

1. Para toda cadeia $w \in L$, $M$ pára e aceita $w$;
2. Para toda cadeia $z \in \Sigma^* - L$, $M$ pára e rejeita $z$ ou executa uma seqüência infinita de movimentações.

Uma linguagem $L$ é dita **estritamente recursivamente enumerável** se, para toda e qualquer Máquina de Turing $M$ que aceita $L$, existir pelo menos uma cadeia $z \in \Sigma^* - L$, tal que $M$ inicie uma seqüência interminável de movimentações em seu processamento.

Da mesma forma que as linguagens recursivas, e diferentemente das linguagens regulares, livres de contexto e sensíveis ao contexto, não se conhece qualquer tipo de estrutura que permita identificar as linguagens recursivamente enumeráveis apenas pela inspeção das propriedades sintáticas de suas sentenças.

As linguagens recursivamente enumeráveis são, portanto, inicialmente caracterizadas a partir das Máquinas de Turing. Conforme será estudado adiante, elas também podem ser caracterizadas por meio do modelo mais geral de gramáticas, as chamadas gramáticas irrestritas (ver Seção 7.4).

A maior importância desta classe de linguagens reside em sua aplicação ao desenvolvimento teórico da computação.

Conforme estudado no Capítulo 6, as linguagens recursivas são também conhecidas como linguagens decidíveis. Isso ocorre porque, para tal classe de linguagens, é sempre possível obter uma Máquina de Turing que sempre pára em resposta a qualquer entrada que lhe seja submetida, não importa se aceitando ou rejeitando tal entrada. Cadeias que pertencem à linguagem são aceitas e cadeias que não lhe pertencem são rejeitadas. Essa característica é particularmente útil quando se deseja estudar a computabilidade de determinados tipos de problema, em particular dos chamados problemas de decisão.

## 7.1 Decidibilidade

Um **problema de decisão** é um problema cuja formulação conduz a apenas duas respostas possíveis: SIM ou NÃO. Exemplo: dadas duas linguagens regulares $L_1$ e $L_2$ quaisquer, é sempre possível determinar se $L_1 = L_2$? Para cada par de linguagens considerado, este problema admite apenas uma resposta entre duas possíveis.

Uma **instância de um problema** é um caso particular de um problema geral, com seus argumentos completamente definidos. Exemplo: dadas a linguagem regular $L_1$ (segue a especificação de $L_1$) e a linguagem regular $L_2$ (segue a especificação de $L_2$), será que $L_1 = L_2$?

Se todas as instâncias de um certo problema, para as quais as respostas forem afirmativas (SIM), forem codificadas sobre um alfabeto $\Sigma$ qualquer, tal conjunto de cadeias forma uma linguagem $L$ sobre $\Sigma$. Note-se que as instâncias de problemas cuja resposta é negativa não pertencem a $L$.

Se a linguagem assim constituída for recursiva, é sabido que existe pelo menos uma Máquina de Turing que decide $L$. Ou seja, dada uma instância qualquer do problema, cuja resposta seja desconhecida, será sempre possível determinar se a resposta é afirmativa ou negativa, bastando para isso verificar se a referida Máquina de Turing aceita ou rejeita a cadeia que representa a instância.

A importância deste resultado está no fato de que linguagens que representam problemas gerais, uma vez identificadas como sendo recursivas, são tais que permitem a determinação mecânica da solução, qualquer que seja a instância considerada. A mecanização da solução, no caso, é implementada pela Máquina de Turing que aceita $L$.

As linguagens recursivamente enumeráveis, por outro lado, não gozam dessa propriedade. Conforme a sua definição, as cadeias não pertencentes à linguagem podem tanto ser rejeitadas após uma parada como provocar a execução de uma seqüência interminável de movimentações na Máquina de Turing correspondente.

Este fenômeno, característico e específico das linguagens estritamente recursivamente enumeráveis, não ocorre com as outras classes de linguagens anteriormente consideradas. De fato, é possível provar que, qualquer que seja a linguagem regular, livre de contexto ou recursiva em questão, é sempre possível obter um autômato finito, um autômato de pilha ou uma Máquina de Turing com fita limitada que sempre pára, qualquer que seja a cadeia de entrada. Isso não significa, no entanto, que não existam reconhecedores desses tipos que eventualmente executem seqüências infinitas de movimentações em resposta a alguma cadeia de entrada.

Considere-se agora um outro problema, cujas instâncias de resposta afirmativa também possam ser codificadas como cadeias sobre um certo alfabeto de entrada. Considere-se ainda que a linguagem formada por essa coleção de cadeias seja do tipo recursivamente enumerável. Isso acarreta a impossibilidade de se verificar, mecanicamente, se determinada cadeia pertence ou não à linguagem, pois cadeias não pertencentes à linguagem eventualmente poderão provocar uma movimentação interminável da Máquina de Turing correspondente.

Logo, torna-se impossível, no caso geral, determinar se um certo problema possui solução, qualquer que seja a instância considerada. Se determinada instância possui resposta SIM (caso em que a cadeia que a representa pertence à linguagem), é fato que a correspondente Máquina de Turing irá parar após um tempo finito de processamento. Se a resposta é NÃO, sabe-se apenas que o processamento poderá eventualmente parar, rejeitando a entrada, ou então iniciar uma seqüência infindável de movimentações.

Para um observador externo, não há nenhuma garantia, para qualquer que seja a instância que venha a ser eventualmente considerada, de que o processamento irá parar em algum momento, produzindo algum resultado.

Devido a essa característica, as linguagens recursivamente enumeráveis são também denominadas **indecidíveis**. Problemas cujas representações na forma de linguagens sejam recursivamente enumeráveis são também chamados de problemas indecidíveis.

De uma forma geral, os termos "recursivo" e "recursivamente enumerável" são empregados quando se trata de linguagens genéricas, e os termos "decidível" (sinônimo de "recursiva") e "indecidível" ou "não-decidível" (sinônimos de "recursivamente enumerável"), quando se trata de linguagens que representam problemas.

O termo **decidibilidade** refere-se ao estudo das linguagens formais, com vistas à determinação das classes a que estas pertencem.

Os termos **solucionável**, **não-solucionável** (ou **insolúvel**) e **parcialmente solucionável**, também empregados quando se trata de problemas, significam, respectivamente, que as correspondentes linguagens são: (i) recursivas, ou seja, podem sempre ser decididas no caso geral; (ii) recursivamente enumeráveis, ou seja, não podem ser decididas no caso geral; e (iii) recursivamente enumeráveis, enfatizando o fato de que pode haver solução para algumas instâncias do problema (ainda que correndo o risco de se esperar indefinidamente por uma resposta).

Aplicada ao estudo dos problemas de decisão, a decidibilidade indica se os mesmos são solucionáveis, não-solucionáveis ou parcialmente solucionáveis.

**Exemplo 7.1** Considere-se o problema $P_V$="Dada uma linguagem regular $L$ qualquer, $L$ é vazia?". Uma linguagem $L_V$ para representar as instâncias desse problema que possuem resposta afirmativa pode ser construída conforme os passos a seguir.

Seja $M = (Q, \Sigma, \delta, q_0, F)$, com $Q = \{q_0, q_1, ...\}$ e $\Sigma = \{\sigma_0, \sigma_1, ...\}$, um autômato finito que define uma linguagem $L$. $M$ pode, por exemplo, ser codificado de forma unívoca na forma de uma cadeia sobre o alfabeto $\{a, b, c, d\}$:

- $q_i \in (Q - F)$ é codificado como $\text{cod}(q_i) = a^{i+1}$;
- $q_i \in F$ é codificado como $\text{cod}(q_i) = a^{i+1}b$;
- $\sigma_i \in \Sigma$ é codificado como $\text{cod}(\sigma_i) = c^{i+1}$;
- $\delta(q_i, \sigma_j) = q_k$ é codificado como $\text{cod}(\delta(q_i, \sigma_j) = q_k) = \text{cod}(q_i)\text{cod}(\sigma_j)\text{cod}(q_k)d$;
- $M$ é codificado como a concatenação das cadeias que representam a codificação das suas transições.

Seguem alguns exemplos de autômatos finitos codificados dessa forma:

- $M_1 = (\{q_0\}, \{x\}, \{(q_0, x) \rightarrow q_0\}, q_0, \varnothing)$:
  $\text{cod}(M_1) = acad$;
- $M_2 = (\{q_0, q_1\}, \{x, y\}, \{(q_0, x) \rightarrow q_1, (q_1, y) \rightarrow q_1\}, q_0, \{q_1\})$:
  $\text{cod}(M_2) = acaabdaabccaabd$;
- $M_3 = (\{q_0, q_1\}, \{x, y\}, \{(q_0, x) \rightarrow q_0, (q_1, y) \rightarrow q_1\}, q_0, \{q_1\})$:
  $\text{cod}(M_3) = acadaabccaabd$;
- $M_4 = (\{q_0, q_1\}, \{x, y\}, \{(q_0, x) \rightarrow q_1, (q_1, y) \rightarrow q_0\}, q_0, \{q_0\})$:
  $\text{cod}(M_4) = abcaadaaccabd$.

A cadeia que representa o autômato $M_2$ possui a estrutura abaixo. As demais cadeias podem ser analisadas da mesma forma.

$$\underbrace{\overbrace{a}^{q_0}\overbrace{c}^{x}\overbrace{aab}^{q_1}d}_{(q_0,x)\rightarrow q_1}\underbrace{\overbrace{aab}^{q_1}\overbrace{cc}^{y}\overbrace{aab}^{q_1}d}_{(q_1,y)\rightarrow q_1}$$

A linguagem $L_V$ que se pretende construir é composta por todas as cadeias codificadas dessa forma, desde que elas representem autômatos finitos que definem a linguagem vazia. Portanto:

- $\text{cod}(M_1) \in L_V$, pois $L(M_1) = \varnothing$;
- $\text{cod}(M_2) \notin L_V$, pois $L(M_2) = xy^*$(e portanto $\neq \varnothing$);
- $\text{cod}(M_3) \in L_V$, pois $L(M_3) = \varnothing$;

- cod($M_4$) $\notin L_V$, pois $L(M_4) = (xy)^*$(e portanto $\neq \varnothing$).

Logo,

$$L_V = \{\text{cod}(M_1), \text{cod}(M_3), ...\} = \{acad, acadaabccaabd, ...\}$$

É possível demonstrar que a linguagem $L_V$ assim construída é recursiva e, portanto, que $P_V$ é decidível. Tal fato pode ser inferido a partir do método (algoritmo) apresentado no Teorema 3.24 da Seção 3.11 para determinar se uma linguagem regular é vazia ou não. Uma Máquina de Turing $N$ que decide $L_V$, após analisar uma cadeia de entrada qualquer $w \in \{a, b, c, d\}^*$, pode ser projetada da seguinte forma:

i. $N$ determina a quantidade de símbolos do alfabeto de entrada do autômato finito codificado ($\left|\Sigma\right|$);

ii. $N$ determina a quantidade de estados do autômato finito codificado ($\left|Q\right|$);

iii. $N$ gera todas as cadeias sobre $\Sigma$ que possuem comprimento maior ou igual a zero e menor que $\left|Q\right|$ (cuja quantidade é $\sum_{i=0}^{|Q|-1} \left|\Sigma\right|^i$);

iv. $N$ simula a operação do autômato $M$ (supondo $w = \text{cod}(M)$) com cada uma das cadeias geradas acima;

v. Caso $M$ não aceite nenhuma dessas cadeias, $N$ pára e aceita $w$; caso $M$ aceite alguma dessas cadeias, ou caso $w$ não represente um autômato finito válido, $N$ pára e rejeita $w$.

Outros sistemas de codificação, diferentes do apresentado neste exemplo, produzem o mesmo resultado, o que torna a escolha de um particular sistema de codificação irrelevante. □

**Exemplo 7.2** Estratégia semelhante à empregada no Exemplo 7.1 pode ser usada para construir linguagens que representam outros problemas de decisão. Por exemplo, pode-se construir uma linguagem $L_A$ para representar o conjunto das cadeias que representam as gramáticas livres de contexto que são ambíguas. Tal linguagem representa, portanto, o problema $P_A$="Dada uma gramática livre de contexto $G$ qualquer, $G$ é ambígua?".

Diferentemente do Exemplo 7.1, é possível demonstrar ([11]) que toda e qualquer codificação que se adote para $L_A$ faz com a mesma resulte estritamente recursivamente enumerável (portanto não-recursiva). Logo, trata-se de um problema que não pode ser decidido no caso geral (conforme antecipado na Seção 4.14). □

É fácil, portanto, justificar o grande interesse prático que existe por problemas para os quais se possa provar que a linguagem de representação é recursiva. Problemas cuja linguagem de representação seja comprovadamente estritamente recursivamente enumerável, por outro lado, servem sobretudo para demonstrar a inexistência de procedimentos mecânicos (algoritmos) que possam resolvê-los no caso geral. E isso é um resultado teórico excepcional, uma vez que evita o desperdício de recursos na busca de soluções gerais e mostra que determinados problemas não possuem solução, independentemente da tecnologia computacional — atual ou futura — que possa ser empregada na busca de pretensas soluções.

Note-se, finalmente, que o fato de um determinado problema poder ser resolvido no caso geral não significa que os algoritmos de resolução conhecidos sejam necessariamente eficientes. Em muitas situações, a busca de solução para o caso geral, cuja existência é conhecida através da teoria, pode ser inviabilizada na prática, em virtude do imenso custo associado à sua realização (custo esse refletido, usualmente, no volume de memória necessário ou no tempo de processamento requerido para se chegar às soluções).

Por outro lado, a inexistência de solução para um problema no caso geral não significa que ele não possa ser resolvido para instâncias específicas ou predeterminadas, ou, ainda, para conjuntos de instâncias predeterminadas do problema original.