UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE CIÊNCIAS NATURAIS E EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA

Filipe Ramos Netto

MÁQUINAS DE TURING, DECIDIBILIDADE E COMPUTABILIDADE

Santa Maria, RS
2024

Filipe Ramos Netto

MÁQUINAS DE TURING, DECIDIBILIDADE E COMPUTABILIDADE

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Matemática, Área de Concentração em Álgebra, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Matemática**.

Orientador: Prof. Pedro Paiva Zühlke D'Oliveira

Santa Maria, RS
2024

Netto, Filipe
Máquinas de Turing, decidibilidade e computabilidade / Filipe Netto.- 2024.
267 p.; 30 cm

Orientador: Pedro Paiva Zühlke D'Oliveira
Dissertação (mestrado) - Universidade Federal de Santa Maria, Centro de Ciências Naturais e Exatas, Programa de Pós-Graduação em Matemática, RS, 2024

1. Álgebra 2. Decidibilidade 3. Máquinas de Turing I. Paiva Zühlke D'Oliveira, Pedro II. Título.

Sistema de geração automática de ficha catalográfica da UFSM. Dados fornecidos pelo autor(a). Sob supervisão da Direção da Divisão de Processos Técnicos da Biblioteca Central. Bibliotecária responsável Paula Schoenfeldt Patta CRB 10/1728.

Declaro, FILIPE NETTO, para os devidos fins e sob as penas da lei, que a pesquisa constante neste trabalho de conclusão de curso (Dissertação) foi por mim elaborada e que as informações necessárias objeto de consulta em literatura e outras fontes estão devidamente referenciadas. Declaro, ainda, que este trabalho ou parte dele não foi apresentado anteriormente para obtenção de qualquer outro grau acadêmico, estando ciente de que a inveracidade da presente declaração poderá resultar na anulação da titulação pela Universidade, entre outras consequências legais.

Filipe Ramos Netto

MÁQUINAS DE TURING, DECIDIBILIDADE E COMPUTABILIDADE

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Matemática, Área de Concentração em Álgebra, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Matemática**.

Aprovado em 16 de fevereiro de 2024:

_____

Pedro Paiva Zühlke D'Oliveira, Dr. (UFSM)
(Presidente/Orientador)

_____

Leonardo Guerini de Souza, Dr. (UFSM)

_____

Oscar Francisco Márquez Sosa, Dr. (UFSC)

_____

Felipe Crivellaro Minuzzi, Dr. (UFSM)

Santa Maria, RS
2024

DEDICATÓRIA

Ao Senhor Jesus, pelas mãos da Virgem Maria. E para que a providência divina faça bom proveito dele, seja a quem ela o levar.

AGRADECIMENTOS

A Deus, Nosso Senhor, que nos sustenta. À Nossa Senhora, mãe de Deus e da Igreja. A São José, patrono dos cristãos. Aos meus pais, Jesus de Barros Netto e Ivete Ramos Netto que estiveram comigo todo este tempo. Ao meu orientador, Doutor Pedro Paiva Zühlke D'Oliveira, pelo acompanhamento nestes semestres. Aos senhores Renaldo Leal da Silva, Ana Maria Alves Tomaz Leal e Ronan Alves Tomaz Leal, por me acolherem em sua casa neste período. Aos professores que tive, aos amigos que fiz em Santa Maria, como também os amigos de Alegrete e do Grupo do Terço.

É próprio do sábio ordenar

(Aristóteles)

RESUMO

MÁQUINAS DE TURING, DECIDIBILIDADE E COMPUTABILIDADE

AUTOR: Filipe Ramos Netto
Orientador: Pedro Paiva Zühlke D'Oliveira

Nessa dissertação consideramos detalhadamente a teoria matemática das máquinas de Turing, que servem como ferramenta de estudo em áreas da Matemática e da Computação. Ao lado delas, lidamos com os tipos de funções que lhe são normalmente associadas, bem como as classificações de linguagens relativas a esse entorno. Trazemos um olhar mais detido a várias demonstrações conhecidas, nas quais expomos uma visão sua em níveis de descrição mais minuciosos. A partir de exemplares de problemas e de situações, verificamos o que significam e o que não significam certos conceitos. Podemos checar isto no trabalho com a noção de computabilidade, e sua não-extensão a certas funções, mesmo que sejam definidas de modo elementar e direto. Apresentamos algumas variantes das máquinas de Turing e também a chamada máquina universal. E concluímos que tais variantes são equivalentes, como modelos de computação, à versão básica das máquinas de Turing. Isto nos permite definir de maneira precisa e independente a noção de algoritmo a partir de tais recursos. Tratamos de diversos problemas a partir de codificações de suas instâncias, muitos deles no próprio universo de tais máquinas. A partir desse método, chegamos a conclusões sobre a capacidade de os resolvermos com as ferramentas em questão, que levantam questões, em certos aspectos, a respeito da existência de um método que universalmente resolva os mesmos problemas. E ainda vemos, através da definição de redutibilidade por mapeamento, que a partir de determinados problemas não-solucionáveis (ou indecidíveis), por meio dos recursos considerados, conseguimos deduzir a incapacidade de encontrar universalmente uma solução para também outros problemas.

Palavras-chave: Máquina de Turing. Decidibilidade. Indecidibilidade. Função computável.

ABSTRACT

TURING MACHINES, DECIDABILITY AND COMPUTABILITY

AUTHOR: Filipe Ramos Netto
ADVISOR: Pedro Paiva Zühlke D'Oliveira

In this dissertation we look in detail at the mathematical theory of Turing machines, which serve as a tool for study in the fields of Mathematics and Computing. Alongside them, we deal with the types of functions that are usually associated with them, as well as the classifications of languages related to this environment. We take a closer look at several well-known demonstrations, in which we present a view of them at more meticulous levels of description. Using examples of problems and situations, we look at what certain concepts mean and don't mean. We can check this in our work with the notion of computability, and its non-extension to certain functions, even if they are defined in an elementary and direct way. We present some variants of the Turing machines and also the so-called universal machine. We conclude that these variants are equivalent, as computing models, to the basic version of Turing machines. This allows us to define the notion of algorithm precisely and independently using these resources. We dealt with various problems by coding their instances, many of them in the universe of such machines. Using this method, we reach conclusions about the ability to solve them with the tools in matter, which raise questions, in certain respects, about the existence of a method that universally solves the same problems. And we also see, through the definition of mapping reducibility, that from certain unsolvable (or undecidable) problems, by means of the resources considered, we can deduce the inability to universally find a solution to other problems as well.

Keywords: Turing machine. Decidability. Undecidability. Computability.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O que traremos neste trabalho é algo que serviu de base para os computadores, as chamadas **máquinas de Turing**. Idealizadas pelo matemático Alan Turing (1937), tratam-se, basicamente, de uma máquina de leitura/escrita sobre uma fita horizontal, sendo preenchida por símbolos conforme um número limitado de comandos.

Essa fita é dividida em quadrados de mesmo tamanho que comportam, cada, um símbolo por vez. O mecanismo que os preenche é a cabeça, que a cada momento opera sobre um determinado quadrado.

Além de imprimir símbolos, a cabeça também escaneia o símbolo que já preenchia o quadrado em que ela está. E tudo isso é governado por um sistema de estados que se sucedem ao longo do trabalho desta máquina. Dependendo do estado em que a máquina de Turing se encontrar, a leitura de determinado símbolo causa a impressão de um símbolo no quadrado atual. Também causa a sucessão a algum outro estado, e causar o movimento da cabeça, que na próxima vez escaneará ou o quadrado adjacente da esquerda ou o da direita.

A fita em questão tem um número infinito de quadrados. Os símbolos que a cabeça pode ler e imprimir, e os estados que a máquina pode assumir são em quantidade finita.

Quando inicia seu trabalho, a máquina de Turing recebe uma entrada limitada na fita, formada alguns destes símbolos. A partir desta entrada e das instruções que a máquina possuir, o resultado da computação de tal máquina será algo de extensão finita ou infinita ao longo da fita. Até porque ela não está limitada pela questão física do tempo.

Definimos precisamente o que são tais máquinas, como funcionam e equivalências nos efeitos de diferentes modos de funcionar no capítulo 3.

Antes disso, o capítulo 2 é dedicado a estabelecer os conceitos numéricos considerados ao longo da dissertação, envolvendo mormente a noção de enumerabilidade.

O conceito de máquinas de Turing foi sendo desenvolvido ao longo dos anos, ganhando um contorno mais algébrico. A partir de certo ponto, também do capítulo 3, começamos a aliar nosso trabalho sobre as máquinas de Turing aos conceitos de alfabeto e de linguagem. Um **alfabeto** é um conjunto finito e não-vazio qualquer. Seus elementos são chamados de **símbolos** do alfabeto. Com eles, formamos as chamadas **cadeias**, que são sequências finitas desses símbolos, podendo haver repetição. As linguagens sobre um alfabeto são nada mais que conjuntos de cadeias advindas dele. Veremos como a utilização desses conceitos facilita muito a consideração e tentativa de resolução de alguns problemas.

No capítulo 4 trazemos algumas variantes na definição de máquina de Turing. Uma delas é constituída de múltiplas fitas para computação. A outra, a partir de uma única fita, se ramifica em várias segundo diversas possibilidades inscritas em suas instruções. Veremos que estas variações razoáveis do modelo original de máquinas de Turing, junto com a original, são, em um sentido a ser esclarecido, todas equivalentes.

Já entre as máquinas de Turing comuns, encontramos um tipo em especial que, a partir das entradas corretas, consegue replicar a computação de outras máquinas de Turing. É a chamada **máquina de Turing universal**, da qual tratamos neste mesmo capítulo.

Já no capítulo 5, temos atrelada às máquinas de Turing a noção de **algoritmo**. Eles serão vistos sob a ótica da resolução de problemas através de nossas máquinas. Tais problemas são tomados a partir de linguagens que os reflitam adequadamente, dada a codificação de suas instâncias por símbolos de algum alfabeto.

Tomamos exemplos de problemas com outras máquinas, os **autômatos finitos**, a fim de verificar a resolução de certos problemas. Os problemas com tais máquinas serão comparados com os problemas análogos envolvendo máquinas de Turing. Dentre estes últimos, temos o *accepting problem* e o *halting problem*, dedicados a conhecer quando máquinas de Turing, sobre certas entradas, respectivamente aceitam sua computação ou simplesmente param em algum momento ao cumputá-las. Só que estes dois problemas são tais que não existe um método universal para verificar quais são e quais não são suas instâncias.

No capítulo 6, nós nos debruçamos sobre resultados encontrados por Shannon (1956) a respeito, primeiramente, de reproduzir a computação de uma máquina de Turing qualquer, mas utilizando pouquíssimos estados. Depois, tratamos da mesma questão, diminuindo, em vez do número de estados, o número de símbolos. Em ambas as situações, acabamos fazendo adaptações necessárias para que os resultados ficassem acomodados com a definição de máquina de Turing que estamos empregando.

No capítulo 7 aproveitamos problemas que antes verificamos não possuírem uma resolução universal para, a partir deles, verificar outros problemas que também não tenham. Veremos que o podemos fazer sob o aspecto de serem **decidíveis**, como também sob o aspecto de serem **Turing-reconhecíveis**. O primeiro diz respeito a computarmos as entradas e chegarmos a um fim sobre todas elas, sinalizando exatamente quais são instãncias do problema e quais não são. O segundo se reduz a sinalizarmos, no fim da computação, as entradas que são instâncias do problema, sem necessariamente finalizar a computação em entradas que não o são.

Formalizamos uma maneira de assim agir, por meio das máquinas de Turing. Introduzimos, então, o conceito de **computabilidade**. Enquanto a decidibilidade diz respeito a linguagens (tomadas usualmente a partir de problemas de interesse), a

computabilidade diz respeito a funções. Lidamos com tais funções em que parte do domínio é normalmente uma linguagem conveniente, sendo o conjunto-imagem dessa parte também conveniente. Para essas funções, cada elemento do domínio é uma entrada de uma máquina de Turing, e suas imagens são as saídas de tal máquina sobre esta entrada. Tal relação está intimamente ligada à questão de problemas cuja solução é alcançável, universalmente falando, por algum algoritmo.

No capítulo 8 exploramos alguns dos limites do conceito de computabilidade no tocante a conjuntos/linguagens, ainda que bem-definidos e finitos. Foi escrito com base em artigo de Radó (1962). Mostramos uma função que cresce mais rapidamente do que qualquer função computável e que, portanto não pode ser computável, ainda que bem-definida. Concomitantemente, expomos uma maneira de converter o conceito de computabilidade do autor para o conceito que havíamos trazido.

Em seguida, no capítulo 9 colocamos algumas convenções e conceitos cuja serventia se dá nos dois capítulos finais. Fixamos um alfabeto conveniente que sirva para imitar a computação de quaisquer outras máquinas de Turing. E apresentamos as chamadas **histórias de computação**, fundamentais na principal demonstração do próximo capítulo.

Nós escrevemos este trabalho motivados pelo alcance que estes recursos têm diante de muitos problemas, sejam eles puramente teóricos ou não. Deste último caso, citamos o problema da correspondência de Post, mostrado no capítulo 10, que se trata, basicamente, de um jogo envolvendo dominós. Embora ele, à primeira vista, não tenha relação com as máquinas de Turing, mostramos que o *halting problem* pode ser reduzido a ele, portanto, ele não é decidível.

O último tópico que trazemos, no capítulo 11 é o teorema da recursão. Suponha que provêssemos a cada máquina de Turing a sua própria descrição como parte de qualquer de suas entradas. Em termos mais concretos, suponha que qualquer programa pudesse ter acesso ao seu próprio código-fonte. De maneira informal, o teorema da recursão afirma que esta propriedade extra nada adiciona ao modelo original. Ou seja, qualquer máquina de Turing com esta capacidade é equivalente a uma máquina de Turing convencional.

Enfim, a utilidade das máquinas de Turing persiste dentro de áreas da Matemática, como a Álgebra de grupos. Em vista de familiarizar o leitor com fundamentos para estas áreas e com o próprio funcionamento das máquinas de Turing, escrevemos tal dissertação.

2 RESULTADOS PRELIMINARES

Em vista de preparar as noções numéricas que utilizaremos ao longo desta dissertação, escrevemos o presente capítulo. Ele trata, basicamente, de conteúdo visto em uma disciplina de Análise Real. Mais especificamente, de como classificar certos conjuntos.

Inicialmente, deixamos claro que, quando nos referimos ao conjunto dos números naturais, está incluído o zero como seu elemento na definição.

Um dado conjunto X é chamado **enumerável** quando ele é finito ou quando existe uma bijeção de $\mathbb{N}$ em X. Consequentemente, um conjunto **não-enumerável** é aquele que não atende nenhuma dessas condições.

Definimos que uma equação de uma variável (variável essa normalmente chamada de x) da forma

$$\sum_{i=0}^{N} a_i x^i = 0 \tag{2.1}$$

com N um inteiro positivo, é chamada **equação polinomial**. Para tal, convém definir que $x^0 = 1$ para qualquer x. Chamamos o número N de **grau** da equação polinomial.

Segundo Petzold (2008), se a_i é inteiro, tal equação é chamada de **equação algébrica**. E as soluções destas equações são chamadas de **números algébricos**. Nesta dissertação, ater-nos-emos aos números algébricos reais. Os números reais não-algébricos são chamados de **números transcendentes**.

Proposição 2.0.1. *O conjunto dos números algébricos contém o conjunto-solução das equações polinomiais onde os coeficientes são racionais.*

Demonstração. Para tal, primeiro vejamos que, como define Lima (2014), o conjunto dos números racionais é dado por $\mathbb{Q} = \left\{ \frac{m}{n};\, m, n \in \mathbb{Z}, n \neq 0 \right\}$. Provaremos que toda solução de uma equação polinomial com grau finito e coeficientes racionais é solução de uma equação algébrica.

Sabemos que, no conjunto dos números reais, uma dada equação do tipo (2.1) tem o mesmo conjunto-solução que a equação

$$\frac{1}{n} \cdot \sum_{i=0}^{N} a_i x^i = \frac{1}{n} \cdot 0 \Leftrightarrow \sum_{i=0}^{N} \frac{a_i}{n} x^i = 0$$

para n um inteiro qualquer. E a última equação acima possui coeficientes racionais, não necessariamente inteiros.

Com isso, encontrar as soluções de uma equação do tipo $\sum_{i=0}^{N} \frac{b_i}{c_i} x^i = 0$, com

$N \in \mathbb{Z}_+^*$, $b_i, c_i \in \mathbb{Z}$ e $c_i \neq 0$, equivale a encontrar as soluções da equação

$$\sum_{i=0}^{N} \left[b_i \left(\frac{1}{c_i} \cdot \prod_{j=0}^{N} c_j \right) \right] x^i = 0.$$

Afinal, ela é uma equação algébrica, uma vez que o fator que está entre parênteses é inteiro (o denominador c_i será cancelado com o fator c_i do produtório). E essa equação, quando tem seus dois membros multiplicados por $1/\left(\prod_{j=0}^{N} c_j\right)$, resulta na equação $\sum_{i=0}^{N} \frac{b_i}{c_i} x^i = 0$. ■

E é fácil provarmos a recíproca desta proposição, pois toda equação algébrica é uma equação polinomial com coeficientes racionais, uma vez que todo inteiro é racional.

Proposição 2.0.2. *O conjunto dos números algébricos está contido no conjunto-solução das equações que possuem a forma que apresentamos em* (2.1)*, mas onde os coeficientes, em vez de inteiros, são racionais.* ■

Dado isso, seja

$$x^N + r_{N-1}x^{N-1} + ... + r_1 x + r_0 = 0 \tag{2.2}$$

uma equação polinomial com coeficientes racionais.

Lema 2.0.3. *Todos os números algébricos são solução de alguma equação do tipo* (2.2)*.*

Demonstração. Se dado número algébrico é solução de uma equação do tipo (2.1), ele também será solução de tal equação com os seus dois membros multiplicados por $1/a_N$. E podemos tomar $r_i = a_i/a_N$. Consideremos r_N o coeficiente de x^N. Consequentemente, os coeficientes do polinômio, que serão cada r_i, serão racionais, e $r_N = 1$. ■

Teorema 2.0.4. *O conjunto dos números algébricos é enumerável.*

Demonstração. Sabemos que o conjunto dos números racionais é infinito e enumerável.

Tomemos então uma bijeção f entre os racionais e os naturais de modo que, para $i \in \mathbb{N}$, o racional r_i é levado em $n_i \in \mathbb{N}$. Uma vez que, como demonstra Lima (2014), o produto cartesiano finito de conjuntos enumeráveis é enumerável, teremos que a N-upla $(r_0, r_1, \ldots, r_{N-1}) \in \mathbb{Q}^N$ será levada pela bijeção f^N (o produto cartesiano de f por ela mesma N vezes) a uma N-upla $(n_0, n_1, \ldots, n_{N-1}) \in \mathbb{N}^N$, sendo tanto o domínio quanto o contradomínio de tal bijeção enumeráveis.

Associemos, agora, cada $(r_0, r_1, \ldots, r_{N-1})$ ao número $M_N \in \mathbb{N}$, dado por $n_0 + n_1 + \cdots + n_{N-1} = M_N$. Vemos que são finitos os elementos de $\mathbb{Q}^N$ associados a cada M_N, pois são finitas as somas de N naturais que totalizam exatamente M_N.

Cada uma das equações polinomiais do tipo (2.2) corresponde a um elemento de $\mathbb{Q}^N$ e vice-versa. E cada uma de tais equações possui no máximo N soluções em $\mathbb{R}$. Assim, dado $M_N \in \mathbb{N}$, existe um número finito $\rho_N(M_N)$ de soluções reais distintas de equações polinomiais do tipo (2.2) satisfazendo $n_0 + n_1 + \cdots + n_{N-1} = M_N$. Isto define uma função ρ_N, de $\mathbb{N}$ em $\mathbb{N}$.

Dado que tais soluções são números algébricos, podemos afirmar então que para cada número natural M_N, existem finitos números algébricos. E, pelo lema 2.0.3, todo número algébrico é solução de alguma equação do tipo (2.2), para algum N inteiro positivo. Segue então que todo número algébrico é um dos $\rho_N(M_N)$ números distintos associados a algum M_N natural.

Como demonstra Lima (2014), a união enumerável de conjuntos enumeráveis é também um conjunto enumerável. Ora, fixando N, para cada valor M_N temos um conjunto V_{M_N} de exatamente $\rho_N(M_N)$ números algébricos associados a ele. Fazendo M_N percorrer o conjunto dos números naturais, e tomando a união de tais conjuntos, obtemos portanto um conjunto enumerável $U_N = \bigcup_{M_N=0}^{+\infty} V_{M_N}$, união enumerável de conjuntos finitos. E o conjunto de todos os números algébricos é exatamente a união $\bigcup_{N=1}^{+\infty} U_N$, e portanto é enumerável, como reunião enumerável de enumeráveis. ■

Observação 1. Como demonstraremos mais à frente nesta dissertação, o conjunto dos números reais é não-enumerável. E como a união de dois enumeráveis é enumerável, só podemos concluir que o conjunto dos números transcendentes também é não-enumerável.

Dito isso, precisando os conceitos, dados um conjunto enumerável (seja ele finito ou infinito) e um conjunto não-enumerável, na relação de ordem entre o "tamanho" dos dois conjuntos, nós dizemos que a cardinalidade do segundo é maior que a do primeiro. Isto quer dizer que mesmo se ao primeiro conjunto nós adicionássemos elementos em quantidade enumerável, ainda assim nós nunca chegaríamos a ter tantos elementos no primeiro conjunto quanto temos no segundo.

Denotamos a cardinalidade de um conjunto Y por $\#Y$. E, sendo X um conjunto enumerável qualquer, quando afirmamos que a cardinalidade de X é menor do que a cardinalidade do conjunto dos reais, isto é, que $\#X < \#\mathbb{R}$, isto significa que existe uma função $f : X \to \mathbb{R}$ injetiva mas não existe uma função $g : \mathbb{R} \to X$ injetiva. Como se, com o conjunto X não conseguíssemos nunca "cobrir" todo o conjunto dos reais.

O fato de que os números reais existem em quantidade não-enumerável significa que a diferença entre tal conjunto e o conjunto dos números algébricos não é só uma diferença de um elemento, ou de dois, ou de três, ou de qualquer quantidade

finita. Também não pode ser de uma quantidade infinita enumerável, pois se o fosse, a união dos algébricos com os transcendentes seria uma união de dois enumeráveis, resultando em outro conjunto enumerável.

Assim, a grosso modo, “existem muitíssimo mais números transcendentes do que algébricos”.

3 MÁQUINAS DE TURING

Uma das primeiras coisas que vamos fazer nesta dissertação é demonstrar que o conjunto dos chamados "números computáveis" é enumerável. Para isso, precisamos definir o que é uma máquina de Turing. Após defini-la, vamos explorar alguns números que podem ser escritos a partir dela. O conceito da máquina de Turing visa ser uma abstração de uma pessoa que efetua contas seguindo um número finito de instruções precisas.

3.1 APRESENTAÇÃO DO CONCEITO

O matemático alemão David Hilbert, após o início do século XX, havia lançado uma série de novos problemas a fim de que os demais matemáticos pudessem encontrar a resposta para eles. Entre tais problemas estava o ***Entscheidungsproblem***, ou **problema de decisão**. Consistia em determinar se, para algum problema arbitrário, sempre havia um algoritmo (finito) capaz de afirmar quais instâncias eram soluções dele e quais não eram.

Para abordar o *Entscheidungsproblem*, Alan Turing, também matemático, escreve um artigo utilizando a ferramenta que denomina, então, como **máquina de Turing**. A partir de tal recurso teórico, ele consegue responder se este problema de Hilbert possui solução.

Aqui mostraremos algo do caminho percorrido até chegar a esta resposta, que se consuma no tratamento do chamado ***halting problem***. Esta parte será concluída no capítulo 5. No capítulo atual exploraremos as máquinas de Turing em sua definição. Descreveremos, pois, nesta seção, um a um de seus elementos constituintes, para daí apresentarmos o conceito da máquina.

Uma máquina de Turing é formada por uma **fita** estendida horizontalmente, dividida em seções chamadas **quadrados**, sendo cada um desses capaz de exibir um **símbolo**. Há infinitos destes quadrados em fila, que convencionalmente são numerados cardinalmente em ordem crescente a partir de um primeiro quadrado escolhido, normalmente da esquerda para a direita. O símbolo exibido no r-ésimo quadrado é denotado por Turing por $\mathfrak{G}(r)$. Na máquina, apenas um quadrado pode, a cada estágio, ser examinado, para que ela interaja a partir do símbolo verificado. O quadrado em questão é chamado de **quadrado escaneado**, e o símbolo presente nele é chamado de **símbolo escaneado**.

A imagem que trazemos mostra uma máquina de Turing durante seu funcionamento. Como podemos ver, neste caso, o símbolo $\mathfrak{G}(1)$ é o símbolo A. Como também,

Figura 1 – Apresentação de uma máquina de Turing

Fonte: Autor (2024)

$\mathfrak{G}(2)$ é H. O símbolo $\mathfrak{G}(3)$ seria, aqui, o espaço vazio, sem um símbolo visível.

Nesta amostra, o quadrado escaneado é o quinto quadrado da fita. E o símbolo escaneado é $\mathfrak{G}(5) =$ E. Vemos tal máquina estar com símbolos até o oitavo quadrado. Logo mais retornaremos a este exemplo.

Em geral, o mecanismo da máquina de Turing que imprime os símbolos nos quadrados e faz as operações que serão apresentadas pode ser chamado de **cabeça** da máquina. Basicamente, é a ação dela que acompanhamos quando vemos a máquina operar.

Cada máquina de Turing é capaz de um número finito de "configurações", usualmente denotadas por $\mathfrak{q}_0, \mathfrak{q}_1, \mathfrak{q}_2, \ldots, \mathfrak{q}_n$ chamadas de **m-configurações** ou **estados**.

Durante sua operação, a máquina sempre se encontra em exatamente um estado q_n e com exatamente um símbolo escaneado $\mathfrak{G}(r)$. Turing chama esse par $(\mathfrak{q}_i, \mathfrak{G}(r))$ de **configuração**. Um estado sempre será sucedido por outro (ou pelo mesmo) após as respectivas operações da máquina a partir dele. A configuração atual é o que determina completamente a operação e estado que vêm imediatamente em sua sequência.

E as chamadas **operações** da máquina, através da cabeça, em geral, consistem em duas.

- **Impressão**: a cabeça é responsável pelos símbolos com que os quadrados escaneados serão preenchidos. Para isso, ela é capaz de duas coisas.
  - Ela pode imprimir um símbolo visível no quadrado atual.
  - Ou ela pode apagar o símbolo que tiver escaneado nele.
- **Deslocamento**: a cabeça também pode mudar o quadrado que está sendo escaneado, desde que o faça deslocando-se um espaço para a esquerda ou para a direita.

Inicialmente, quando indicarmos em tabelas tais operações da cabeça, o faremos por abreviações. A operação de imprimir um símbolo α será abreviada para Pα (por causa de *print*/imprimir). A operação de apagar será abreviada para E (por causa de *erase*/apagar). O deslocamento para a direita será abreviado para R (de *right*/direita). E o deslocamento para a esquerda, para L (de *left*/esquerda).

Colecionamos as operações de apagar e de imprimir como participantes de uma mesma natureza. Pois, como faremos mais à frente, trataremos a operação de apagar como equivalente à operação de "imprimir espaço vazio".

Exemplo 1. Voltando à máquina ilustrada, com a cabeça sobre o quinto quadrado, ela escaneava o símbolo E. Digamos que, neste momento a máquina se encontrasse no estado $\mathfrak{q}_2$, e que a configuração $(\mathfrak{q}_2, \mathsf{E})$ cause a impressão do símbolo C, o deslocamento da cabeça para a direita e a sucessão para o estado $\mathfrak{q}_4$.

Figura 2 – Seguimento do exemplo (parte 1)

Fonte: Autor (2024)

Temos um novo quadrado escaneado, o sexto, e o símbolo escaneado é J. Digamos, pois, que a configuração $(\mathfrak{q}_4, \mathsf{J})$ cause o apagamento do símbolo no quadrado, o deslocamento da cabeça para a direita e a sucessão para o estado $\mathfrak{q}_2$.

Figura 3 – Seguimento do exemplo (parte 2)

Fonte: Autor (2024)

O quadrado escaneado atual é o sétimo, sendo C o símbolo escaneado. Esta máquina poderia ser tal que, sob a configuração $(\mathfrak{q}_2, \mathsf{C})$, operar apenas o deslocamento para a esquerda, sem haver a impressão de um novo símbolo, e ter a sucessão para o estado $\mathfrak{q}_5$.

Figura 4 – Seguimento do exemplo (parte 3)

Fonte: Autor (2024)

Esta última parte serve para vermos que um mesmo estado, como o $\mathfrak{q}_2$, acompanhado de diferentes símbolos atuais, pode vir a causar diferentes ações na máquina. É o par da configuração que determina o que virá a seguir.

Tal máquina continuaria trabalhando, onde a próxima configuração seria a do estado $\mathfrak{q}_5$ tendo o espaço vazio como símbolo acompanhante. Não é diferente do que já vinha acontecendo. Podemos supor que isso causaria as operações de impressão de H, o deslocamento para a direita e a sucessão para o estado $\mathfrak{q}_1$. E depois haverá tantas outras configurações mais.

Figura 5 – Seguimento do exemplo (parte 4)

Fonte: Autor (2024)

□

Após este vislumbre dentro do trabalho de uma máquina de Turing, a partir das descrições dadas pelo autor, vamos precisar o seu conceito.

Definição 1. É uma **máquina de Turing** a instância composta por:

- uma fita horizontal de infinitos quadrados lado a lado, sendo um deles o inicial;
- um conjunto finito de símbolos passíveis de leitura e impressão;
- um conjunto finito de estados passíveis de, um por vez, presença durante a atividade de máquina;
- uma cabeça leitora e impressora, que começa sobre o quadrado inicial;
- um estado inicial;
- um sistema de regras onde, para cada configuração, ocorrem, ou respectivamente uma operação de impressão e uma operação de deslocamento, ou apenas uma destas duas, ou então nenhuma delas;
- pelo mesmo sistema de regras, imediatamente após as operações causadas por uma configuração, ocorre uma sucessão de estado;

de modo que a fita, ou começa com nenhum quadrado preenchido, ou, naquela que chamaremos posteriormente de **máquina de Turing universal**, possui um número finito de quadrados preenchidos em seu início, intercalados com alguns quadrados vazios.

Rigorosamente falando, as máquinas de Turing não “se lembram” do símbolo escaneado antecessor depois que trocam de quadrado. No entanto podemos condicionar certas operações da máquina, em sequência, a depender do símbolo escaneado previamente e/ou do antigo estado. Essa afirmação faz sentido quando, como veremos à frente, inserimos certas funções, das quais os estados e símbolos serão argumentos.

Inicialmente, pensemos que o objetivo de uma tal máquina é nos fornecer um determinado número a partir da impressão de seus dígitos em base binária em parte da fita (digamos, nos quadrados de índices pares), sendo os algarismos 0 e 1 os símbolos que estarão nos quadrados relevantes. Tal máquina inicia com uma fita em branco, isto é, tal que seus quadrados estejam sem quaisquer símbolos visíveis. Suponha, ainda, que seja fornecida uma tabela que indique todos as suas possíveis configurações e, para cada uma delas, a operação e estados subsequentes. Quando ela é posta em ação, a partir de um estado inicial e tabela dadas, a sequência de 0’s e 1’s impressa nos quadrados relevantes será chamada de **sequência computada**

pela máquina. E o número real (entre 0 e 1) cuja parte fracionária é representada, na base binária, por esta sequência, é chamado de **número computado pela máquina**.

Ainda, quando nos referirmos a "estágio" dentro da atividade da máquina, trata-se de um "momento" em que a máquina estaria colocando em ato algum dos determinados passos para os quais ela foi construída. Contudo, este momento existe apenas em ideia, e não fisicamente.

Imaginemos uma máquina de Turing que fosse transposta para a realidade com todas suas condições. E que ela produzisse, por exemplo, a sequência, em base binária, da dízima periódica $0,100100100\ldots$ Quando cada configuração fosse posta em ato, a máquina teria um intervalo de tempo sendo usado para tal. E a duração deste intervalo seria limitada inferiormente por questões físicas. Consequentemente, a máquina de Turing ficaria preenchendo a fita com a sequência em questão para sempre, o que não seria conveniente.

Aliás, o número utilizado acima, em base decimal, é

$$2^{-1} + 2^{-4} + 2^{-7} + 2^{-10} + \ldots$$

$$= 2^{-1} + 2^{-1} \cdot 2^{-3} + 2^{-1} \cdot (2^{-3})^2 + 2^{-1} \cdot (2^{-3})^3 + \ldots$$

$$= \frac{2^{-1}}{1 - 2^{-3}} = \frac{1/2}{1 - 1/8} = \frac{1/2}{7/8} = \frac{4}{7}.$$

Continuando a descrição das máquinas de Turing, há dois tipos de símbolos que elas imprimem. Os símbolos do primeiro tipo se encontram nos chamados **F-quadrados**. Os referidos símbolos normalmente consistem de algarismos, tais como 0 e 1, para representar números computáveis na base binária. O "F" vem da palavra *figure*, que significa "algarismo". E os símbolos do primeiro tipo são chamados de **símbolos permanentes**, ou ainda, de **F-símbolos**. Os outros símbolos se encontram nos chamados **E-quadrados** (devido à palavra *erasable*, apagável), e podem ser chamados de **símbolos apagáveis**, ou ainda, de **E-símbolos**. Será nos F-quadrados que estarão os símbolos relevantes para a sequência computada ao final.

Assim chamamos estes dois tipos de símbolos pois, quando um F-símbolo é impresso em um F-quadrado, ele nunca será apagado dele. O que não acontece com os E-símbolos. Fica estabelecido, para as máquinas de Turing, que, quando um F-quadrado deixa de estar em branco, ele ficará com o novo símbolo durante todo o decorrer da atividade da máquina. E também fica estabelecido que os F-quadrados serão preenchidos sempre do atual para seu sucessor. Nunca um ou mais deles ficarão vazios entre dois F-quadrados preenchidos.

Em cada estágio da operação da máquina, chamamos de **configuração completa** daquele estágio o conjunto formado pelo número do quadrado escaneado, pela sequência completa de todos os símbolos na fita, e pelo estado atuais. E as mudan-

ças entre as sucessivas configurações completas da máquina e fita serão chamadas de **transições** da máquina.

Dito isso, vejamos a computação de uma sequência relativa a um número real na base binária, justamente o $\frac{4}{7}$.

Exemplo 2. Apresentaremos o funcionamento de uma máquina de Turing, a qual produzirá justamente a sequência 100100100... Convencionamos que os F-quadrados, a partir do primeiro deles, o sejam exatamente de dois em dois, da esquerda para a direita. Faremos nossa descrição por meio de uma tabela que aponta as transições a partir das configurações, e por meio da informação de qual é o estado inicial.

Tal tabela possui quatro colunas, e cada linha traz o próximo estado e as operações a serem realizadas. A primeira coluna traz o estado de uma possível configuração atual. A segunda coluna indica o símbolo escaneado. A terceira, as operações a serem praticadas pela máquina. E a quarta, o próximo estado da máquina. Os estados foram listados na ordem alfabética, da letra “a” até a letra “i”.

Tabela 1 – Descrição tabular da máquina

Estado	**Símbolo**	**Operações**	**Estado final**
$\mathfrak{a}$	Nada	P1, R	$\mathfrak{b}$
$\mathfrak{b}$	Nada	P$\mathbf{u}$	$\mathfrak{c}$
$\mathfrak{c}$	$\mathbf{u}$	E, R	$\mathfrak{d}$
$\mathfrak{d}$	Nada	P0, R	$\mathfrak{e}$
$\mathfrak{e}$	Nada	P$\mathbf{v}$	$\mathfrak{f}$
$\mathfrak{f}$	$\mathbf{v}$	E, R	$\mathfrak{g}$
$\mathfrak{g}$	Nada	P0, R	$\mathfrak{h}$
$\mathfrak{h}$	Nada	P$\mathbf{w}$	$\mathfrak{i}$
$\mathfrak{i}$	$\mathbf{w}$	E, R	$\mathfrak{a}$

Fonte: Autor (2024)

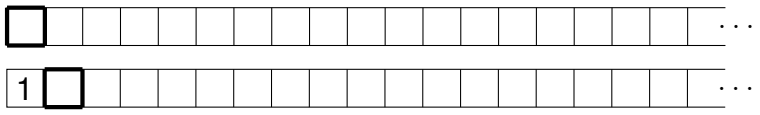
O estado inicial desta máquina será o estado $\mathfrak{a}$. Observemos como será o trabalho dessa máquina. Podemos ver que, dado o seu início em um quadrado qualquer da fita, ela imprimirá os F-símbolos de dois em dois quadrados, da esquerda para a direita. A partir do primeiro F-quadrado, será também F-quadrado todo e unicamente aquele quadrado que estiver a um número par de quadrados à direita daquele primeiro. Consequentemente, o quadrado exatamente entre dois F-quadrados seguidos será um E-quadrado.

Começando no primeiro quadrado, a máquina, no estado $\mathfrak{a}$ e escaneando um quadrado vazio, imprime o F-símbolo 1 e anda uma unidade para a direita.

Olhemos para a questão da configuração completa, no primeiro estágio, o qual está na figura 6 representado. Ela consiste na informação de a fita estar vazia de símbolos, com o quadrado escaneado sendo o primeiro, e a máquina sob o estado $\mathfrak{a}$.

No segundo estágio, a configuração completa consiste no símbolo 1 apenas,

Figura 6 – Primeiro exemplar de computação de uma sequência binária (parte 1)

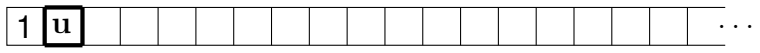


Fonte: Autor (2024)

no primeiro quadrado, com o quadrado escaneado sendo o segundo, e a máquina, segundo a tabela, sob o estado $\mathfrak{b}$.

Agora, então no estado $\mathfrak{b}$ e escanenado um quadrado vazio, a máquina vai imprimir o E-símbolo $\mathfrak{u}$ no quadrado escaneado.

Figura 7 – Primeiro exemplar de computação (parte 2)



Fonte: Autor (2024)

Aqui a configuração completa já consiste na sequência de símbolos 1$\mathfrak{u}$, com o quadrado escaneado sendo o segundo, e a máquina sob o estado $\mathfrak{c}$.

Daí, no estado $\mathfrak{c}$ e escaneando o símbolo $\mathfrak{u}$, a máquina apaga o símbolo do E-quadrado escaneado, e anda uma unidade para a direita.

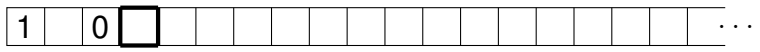
Figura 8 – Primeiro exemplar de computação (parte 3)

Fonte: Autor (2024)

A configuração completa, neste estágio, consiste no símbolo 1, com o quadrado escaneado sendo o terceiro, e a máquina sob o estado $\mathfrak{d}$.

Assim, no estado $\mathfrak{d}$ e escaneando um quadrado vazio, a máquina imprime o F-símbolo 0 no quadrado escaneado, e anda uma unidade para a direita.

Figura 9 – Primeiro exemplar de computação (parte 4)

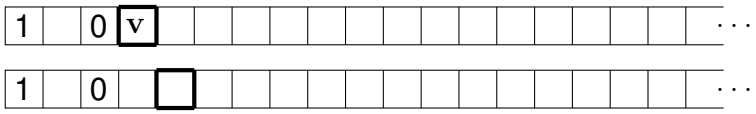


Fonte: Autor (2024)

Para falarmos da atual configuração completa, usemos o caracter ␣ para representar o “símbolo vazio”, onde não há símbolo visível. Pois que, neste estágio, ela consiste na sequência 1␣0, com o quadrado escaneado sendo o terceiro, e a máquina sob o estado $\mathfrak{e}$.

No estado $\mathfrak{e}$, a máquina imprime o E-símbolo $\mathfrak{v}$ no quadrado escaneado. E então no estado $\mathfrak{f}$ ela o apaga e anda uma unidade para a direita.

Figura 10 – Primeiro exemplar de computação (parte 5)

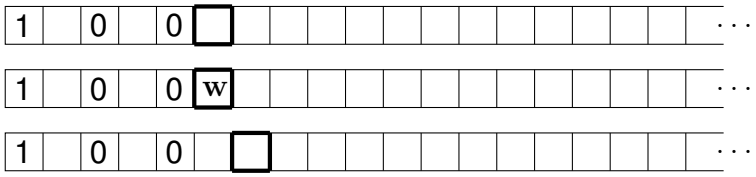


Fonte: Autor (2024)

A esta altura, já pudemos compreender como são as configurações completas de cada estágio de uma máquina de Turing. E, após essas duas transições, no atual estágio, a máquina se encontra sob o estado $\mathfrak{g}$.

Do estado $\mathfrak{g}$ até o estado $\mathfrak{i}$, a máquina age com padrão semelhante aos passos antes vistos. Dessa vez imprime o F-símbolo 0, anda para o quadrado da direita, onde imprime o E-símbolo $\mathfrak{w}$, o apaga e anda mais uma vez para a direita.

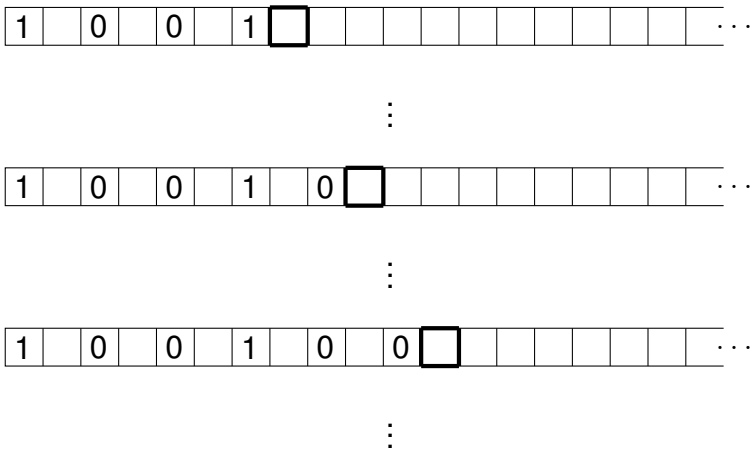
Figura 11 – Primeiro exemplar de computação (parte 6)



Fonte: Autor (2024)

Finalmente, a máquina retorna para o estado $\mathfrak{a}$, o mesmo estado do início. E encontrando-se em $\mathfrak{a}$, a máquina escaneará novamente um quadrado vazio, estando diante de mais quadrados vazios à sua direita. Acontecerá, portanto, a repetição do mesmo conjunto de ações, produzindo as sequências “100” restantes. A máquina prossegue indefinidamente no preenchimento de tais sequências.

Figura 12 – Primeiro exemplar de computação (parte 7)



Fonte: Autor (2024)

Este é um exemplo relativamente básico para termos noção de como uma máquina de Turing trabalha. Com finitos estados, ela conseguiu computar uma sequência infinita.

Na situação descrita, os diferentes estados ficaram indicados, cada um, de modo que neles o escaneamento é sempre de um símbolo específico. E é de fato o que está acontecendo na máquina que montamos. Mas não precisamos de uma quantidade tão grande de estados para tal tarefa.

É perceptível que alguns estados da tabela são muito semelhantes entre si. Vemos isto com os estados $\mathfrak{c}$, $\mathfrak{f}$ e $\mathfrak{i}$, a partir dos quais a máquina realiza a mesma sequência de operações, com diferentes símbolos escaneados em cada um dos casos. E em seguida, ela passará para os estados com os quais imprimirá o próximo F-símbolo da sequência.

Visto que tais estados agem a partir de símbolos escaneados distintos, podemos organizar a tabela de maneira mais compacta. Em vez de termos 9 estados, teremos apenas 7. Tal como na tabela a seguir.

Tabela 2 – Descrição tabular compacta da máquina

Estado	Símbolo	Operações	Estado final
$\mathfrak{a}$	Nada	P1, R	$\mathfrak{b}$
$\mathfrak{b}$	Nada	P$\mathbf{u}$	$\mathfrak{c}$
$\mathfrak{c}$	$\mathbf{u}$	E, R	$\mathfrak{d}$
	$\mathbf{v}$	E, R	$\mathfrak{g}$
	$\mathbf{w}$	E, R	$\mathfrak{a}$
$\mathfrak{d}$	Nada	P0, R	$\mathfrak{e}$
$\mathfrak{e}$	Nada	P$\mathbf{v}$	$\mathfrak{c}$
$\mathfrak{g}$	Nada	P0, R	$\mathfrak{h}$
$\mathfrak{h}$	Nada	P$\mathbf{w}$	$\mathfrak{c}$

Fonte: Autor (2024)

Iniciando sua atividade no estado $\mathfrak{a}$, a máquina assim descrita agirá essencialmente do mesmo modo que antes. No entanto, sempre que ela passar para um estado que escaneará o E-símbolo da vez, esse estado não será um entre três, mas sempre o estado $\mathfrak{c}$. Ele admitirá diferentes comportamentos, dependendo de qual é o E-símbolo escaneado. Neste caso, o que muda realmente é apenas o próximo estado para o qual a máquina passará, pois as operações são as mesmas para os três E-símbolos.

Nesta situação, os F-quadrados foram os quadrados de numeração ímpar, desde o primeiro, passando pelo terceiro, quinto, sétimo, em diante. Os de numeração par foram os E-quadrados. □

Tal como neste exemplo, convencionaremos que o estado inicial é sempre aquele que aparece na primeira coluna da primeira linha da tabela.

Também fica convencionado chamar **marcadores** os E-símbolos desenhados

um quadrado à direita de um F-quadrado já preenchido com um F-símbolo. Falamos que, se o F-símbolo d está com o marcador $\mathbf{y}$ à sua direita, então d está marcado por $\mathbf{y}$, e que $\mathbf{y}$ é marcador de d.

Neste caso, tivemos o símbolo $\mathbf{u}$ como marcador de um símbolo 1, o símbolo $\mathbf{v}$ como marcador de um 0, e o símbolo $\mathbf{w}$ como marcador também de um 0. Em verdade, poderíamos ter feito trabalho semelhante ao do exemplo sem estes marcadores. Eles se verificam mais úteis, porém, em outros contextos. Um deles se dá na máquina de Turing universal, o que veremos no próximo capítulo.

Destacamos, agora, as definições de **máquina circular**, que é a máquina de Turing que nunca escreve mais do que uma quantidade finita de F-símbolos ao longo da fita (ou seja, cujas sequências computadas são sempre finitas), e de **máquina não-circular**, que se trata da máquina de Turing que não é circular. Uma máquina circular pode ser tal que parem as operações em algum momento, imprimindo símbolos até um dado quadrado apenas. Ou mesmo uma máquina que imprima infinitos símbolos, mas apenas nos E-quadrados, com finitos símbolos nos F-quadrados.

A máquina de Turing do exemplo acima se tratava de uma máquina não-circular. Ela imprime a sequência infinita de 100100100... através dos F-quadrados.

Agora, para uso em seguida, definamos a função $\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{Z}$, chamada função chão. Para todo x real, $\lfloor x \rfloor = m$, onde $m \leq x < m + 1$.

Dado isso, para todo número real x, existe um único valor $t \in \mathbb{R}$, $0 \leq t < 1$, tal que $x = \lfloor x \rfloor + t$. Chamamos t de **parte fracionária** de x.

Com isso em mente, a partir de Turing (1937), tomamos outras duas definições. Uma **sequência computável** é uma sequência que pode ser computada por uma máquina não-circular. E ainda, havemos de fazer a distinção entre sequências computáveis e números computáveis. Um **número computável** é um número real cuja parte fracionária pode ser obtida por meio de uma máquina de Turing não-circular. Ou seja, a parte fracionária de um número computável equivale a uma sequência computável.

Podemos lembrar que os números computáveis podem ter finitas ou infinitas casas decimais não nulas. Ao falar isso, o objetivo é notarmos que mesmo números com expansão finita, como $0,75 = 0,5 + 0,25 = 2^{-1} + 2^{-2}$, que na base binária é representado por $0,11$, podem ser escritos com infinitas casas decimais, o qual no caso seria $0,11000\ldots$, com apenas zeros após o segundo "1". E assim, vemos que tal número poderia ser escrito, nesta última forma, em uma máquina não-circular. E também poderia ser escrito, como expansão finita, em uma máquina circular.

No entanto, as máquinas circulares não são consideradas como as máquinas "boas", por assim dizer, pois não conseguiriam ter uma sequência computada relativa a um número que só possui expansão infinita. Como é o caso do $\frac{4}{7}$ na base binária.

Como pudemos ver, este número é computável. Ou seja, ele tem sua sequência de dígitos computada por uma máquina não-circular. Mas não a poderia ter por

uma máquina circular, já que ele exigiria, na base binária, a impressão em infinitos F-quadrados. E por isso, consideramos como as máquinas “boas”, as não-circulares. Elas abarcam sequências relativas a números que têm expansão finita, como também a números que não têm. Com isso, conseguimos afirmar que todo número com expansão finita é computável.

3.2 SEQUÊNCIAS COMPUTÁVEIS

Há observações que aqui podemos fazer, a partir do que demonstram Hämmerlin e Hoffmann (1991) a respeito de todos os números reais terem uma representação na base binária.

Teorema 3.2.1. *Consideremos uma fração $\frac{a}{b}$ irredutível, com $a \in \mathbb{N}$ e $b \in \mathbb{N}^*$ na base decimal. Então um número, na parte fracionária, possui, na base binária, expansão:*

a) *finita se, e somente se ele é um racional da forma acima, com b igual a uma potência de base 2 e expoente natural;*

b) *apenas infinita e periódica se, e somente se ele é um racional da forma acima, com b diferente de uma potência de base 2 e expoente natural.*

c) *infinita e não-periódica se, e somente se ele é um número irracional.*

Cabe aqui destacarmos que, neste item, estamos tratando dos números cuja expansão na parte fracionária é **apenas** infinita e periódica, pois alguns números podem ser escritos com uma forma em expansão finita e com outra em expansão infinita, e isso excluindo o caso de um período composto apenas por algarismos zero, que aqui está sendo desconsiderado.

Tomemos por exemplo, o número $x = 0,1101111\ldots$ na base binária, onde o “1” se repete de forma ininterrupta e infinita. A parte relativa a este período equivale a

$$\sum_{i=0}^{+\infty} \left(1 \cdot 2^{-4}\right) \cdot 2^{-i} = \sum_{i=0}^{+\infty} 2^{-4} \cdot \left(2^{-1}\right)^i = \frac{2^{-4}}{1 - 2^{-1}} = \frac{1/16}{1 - 1/2} = \frac{1/16}{1/2} = \frac{1}{8} = 2^{-3},$$

valor representado por $0,001$ na base binária. Portanto, o mesmo número pode ser representado por $0,101111\ldots$ e por $0,111$.

O número x, assim, não entra no caso descrito pelo item **b)**. Entram no caso de **b)** números que, na base binária, só são representados por expansão infinita.

Enfim, este teorema é de conhecimento relativamente generalizado, e portanto não traremos sua demonstração.

Corolário 3.2.2. *Todo número racional possui parte fracionária, na base binária, cuja representação é finita, ou cuja representação é infinita e periódica.* ■

Deixaremos aqui alguns dos elementos que depois utilizaremos. Adotaremos as notações a seguir.

- A representação $(t_{M_1} \dots t_2 t_1 t_0, t_{-1} t_{-2} \dots t_{-M_2})_b$ indica o número

$$t_{M_1} \cdot b^{M_1} + \cdots + t_2 \cdot b^2 + t_1 \cdot b + t_0 + t_{-1} \cdot b^{-1} + \cdots + t_{-M_2} \cdot b^{-M_2},$$

  onde $M_1 \geq 0$, $M_2 \geq 1$ e $b \geq 2$ são inteiros e, para $i \in \mathbb{Z}$, temos que $t_i \in \mathbb{Z}$, chamado algarismo, é tal que $0 \leq t_i < b$. A potência de base b por que cada algarismo está multiplicado depende de sua posição em torno da vírgula. Se t_i está na k-ésima posição à esquerda da vírgula, então ele é multiplicado por b^{k-1}, e se ele está na k-ésima posição à direita da vírgula, então ele é multiplicado por b^{-k}.

- E a representação $\left(t_{M_1} \dots t_2 t_1 t_0, t_{-1} t_{-2} \dots t_{-r} \overline{t_{-(r+1)} t_{-(r+2)} \dots t_{-(r+s)}}\right)_b$ indica o número

$$\begin{aligned} & t_{M_1} \cdot b^{M_1} + \cdots + t_2 \cdot b^2 + t_1 \cdot b + t_0 + t_{-1} \cdot b^{-1} + \dots \\ & + t_{-r} \cdot b^{-r} + \sum_{j=0}^{+\infty} \Bigg(t_{-(r+1)} \cdot b^{-(r+1)} + t_{-(r+2)} \cdot b^{-(r+2)} \\ & \qquad + \cdots + t_{-(r+s)} \cdot b^{-(r+s)} \Bigg) \cdot b^{-(j \cdot s)}, \end{aligned}$$

  onde $r, s \in \mathbb{N}$, e para $i \in \mathbb{Z}$, os t_i são ainda algarismos, com as mesmas propriedades. Para $\left(t_{M_1} \dots t_2 t_1 t_0, \overline{t_{-1} t_{-2} \dots t_{-s}}\right)_b$, analogamente ao caso anterior, não há outras parcelas com fator de potências de base b e expoente negativo antes do somatório. Nós denominamos a parte dos algarismos sobretachada, isto é, a sequência $t_{-(r+1)} t_{-(r+2)} \dots t_{-(r+s)}$, (ou então $t_{-1} t_{-2} \dots t_{-s}$), como o período da dízima em questão, e dizemos que o comprimento desse período é s.

  Podemos reescrever essas representações como respectivamente equivalentes a

$$\left(t_{M_1} \dots t_0, t_{-1} \dots t_{-r} t_{-(r+1)} \dots t_{-(r+s)} t_{-(r+1)} \dots t_{-(r+s)} \dots\right)_b$$

  e a

$$\left(t_{M_1} \dots t_0, t_{-1} \dots t_{-s} t_{-1} \dots t_{-s} \dots\right)_b,$$

  nas quais está significado que os algarismos do período vão se repetindo ordenadamente e indefinidamente, acompanhando as potências binárias das posições em que estiverem.

Exemplo 3. Com essa notação, escrevemos que

$$\frac{4}{7} = (0,571428571428\ldots)_{10} = \left(0,\overline{571428}\right)_{10} = \left(0,\overline{100}\right)_{2} = (0,100100\ldots)_{2}.$$

□

Exemplo 4. Verificamos que o número $0,10100110011001\ldots$, na base binária, com período "1001", é igual a

$$2^{-1} + \left(2^{-3} + 2^{-6}\right) + \left(2^{-7} + 2^{-10}\right) + \left(2^{-11} + 2^{-14}\right) + \ldots$$

$$= 2^{-1} + \left(2^{-3} + 2^{-6}\right) + \left(2^{-3} + 2^{-6}\right) \cdot 2^{-4} + \left(2^{-3} + 2^{-6}\right) \cdot \left(2^{-4}\right)^2 + \ldots$$

$$= 2^{-1} + \frac{2^{-3} + 2^{-6}}{1 - 2^{-4}} = 2^{-1} + \frac{1/8 + 1/64}{1 - 1/16} = \frac{1}{2} + \frac{9/64}{15/16} = \frac{1}{2} + \frac{3}{20} = \frac{13}{20}.$$

Com a presente notação, escrevemos que

$$\frac{13}{20} = (0,65)_{10} = \left(0,10\overline{1001}\right)_{2}.$$

□

Agora, acrescentando à nossa descrição algo que Turing adota, escreveremos cada sequência na máquina de modo que o primeiro F-símbolo é acompanhado de dois símbolos ə em seus dois quadrados imediatamente predecessores. Esse símbolo é chamado "schwa". O autor o posicionou assim a fim de que houvesse um símbolo que servisse exatamente para sinalizar onde ficava o início da fita.

Com este recurso, quando a cabeça se deparar com o ə, não tentará avançar inutilmente para antes do começo. Para isso, claro, é necessário deixar preparadas as instruções correlatas nas transições das configurações.

Assim, tomando a sequência computada no exemplo da seção anterior, relativa ao número $\left(0,\overline{100}\right)_{2}$, podemos, a partir da tabela feita originalmente (da versão compacta), formar uma nova tabela que atenda a esta condição de a fita iniciar com os símbolos ə.

A fita desta nova máquina de Turing, após feita toda a computação, fica como representamos abaixo.

Figura 13 – Máquina anterior atualizada e após a computação

ə	ə	1		0		0		1		0		0		1		0		0	⋯

Fonte: Autor (2024)

Para tal máquina de Turing, os F-quadrados são os quadrados de número ímpar ainda, só que, dessa vez, do terceiro em diante. E os E-quadrados são os quadrados de número par, só que do quarto em diante.

Tabela 3 – Descrição tabular atualizada da máquina

Estado	Símbolo	Operações	Estado final
j	Nada	Pə, R	ɬ
ɬ	Nada	Pə, R	a
a	Nada	P1, R	b
b	Nada	P**u**	c
c	**u**	E, R	d
	v	E, R	g
	w	E, R	a
d	Nada	P0, R	e
e	Nada	P**v**	c
g	Nada	P0, R	h
h	Nada	P**w**	c

Fonte: Autor (2024)

A partir do número representado no exemplo 4, vejamos uma máquina de Turing, já com a descrição atualizada.

Exemplo 5. Apresentaremos o funcionamento de uma máquina de Turing, a qual produzirá justamente a sequência 10100110011001..., relativa ao número $\frac{13}{20}$.

Tabela 4 – Descrição tabular da máquina atual

Estado	Símbolo	Operações	Estado final
a	Nada	Pə, R	b
b	Nada	Pə, R	c
c	Nada	P1, R	d
d	Nada	R	e
e	Nada	P0, R	f
f	Nada	R	g
g	Nada	P1, R	h
h	Nada	R	i
i	Nada	P0, R	j
j	Nada	R	ɬ
ɬ	Nada	P0, R	l
l	Nada	R	m
m	Nada	P0, R	f

Fonte: Autor (2024)

Figura 14 – Máquina atual após a computação

ə	ə	1		0		1		0		0		1		1		0		0		1	···

Fonte: Autor (2024)

Como convencionamos, começando a atividade da máquina pelo estado da

primeira linha da tabela, iniciamos em ɑ. Aqui optamos por não usar marcadores. Imprimimos, fora os ə, apenas F-símbolos. Os E-quadrados ficarão vazios.

É fácil vermos que, após a computação, a fita de tal máquina de Turing fica como exposto acima. □

Proposição 3.2.3. *Todo número racional é um número computável.*

Demonstração. Segundo o corolário 3.2.2, a parte fracionária de todo número racional, na base binária, tem expansão finita ou tem expansão infinita e periódica. Então basta que nós provemos que estes dois casos correspondem a números computáveis. Se tal parte fracionária tiver expansão finita, então, para algum $m \geq 1$ natural **i)** ela é representada por $(0, d_1 d_2 \dots d_m)_2$, também podendo ser representada por

$$(0, d_1 d_2 \dots d_m 0000 \dots)_2$$

com $d_1, \dots, d_m$ algarismos.

Se o presente número racional tiver expansão infinita e periódica, então, para algum $m \geq 1$ e algum $n \geq 1$ naturais, **ii)** ou ele possui a parte fracionária representada por $(0, d_1 d_2 \dots d_m \overline{e_1 e_2 \dots e_n})_2$, também podendo ser representada por

$$(0, d_1 d_2 \dots d_m e_1 e_2 \dots e_n e_1 e_2 \dots e_n \dots)_2$$

com $d_1, d_2, \dots, d_m, e_1, e_2, \dots, e_n$ algarismos; **iii)** ou ele possui a parte fracionária representada por $(0, \overline{e_1 e_2 \dots e_n})_2$, também podendo ser representada por

$$(0, e_1 e_2 \dots e_n e_1 e_2 \dots e_n \dots)_2$$

com $e_1, e_2, \dots, e_n$ algarismos.

As máquinas de Turing tomadas representarão infinitos dígitos, usando infinitos F-quadrados, sem uma notação que indique a repetição de algarismos. Então cada número racional terá sua sequência computada na fita conforme sempre a última representação de cada um dos três casos mostrados acima.

i) Se $m = 1$, então fica evidente que a partir de instruções como da tabela 5, trazendo 5 estados, damos conta da sequência a ser computada.

Assim, com muita facilidade, conseguimos produzir, na fita, a sequência tal como indicada na figura 15.

Para $m > 1$, podemos utilizar a tabela 6, trazendo $(2m + 4)$ estados.

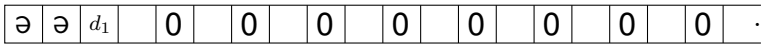
A máquina de Turing que montamos a partir desta tabela produzirá a sequência na fita tal como mostrada na figura 16.

Tabela 5 – Descrição da máquina para o primeiro caso, com $m = 1$

Estado	Símbolo	Operações	Estado final
$\mathfrak{a}$	Nada	Pə, R	$\mathfrak{b}$
$\mathfrak{b}$	Nada	Pə, R	$\mathfrak{c}$
$\mathfrak{c}$	Nada	Pd_1, R	$\mathfrak{d}$
$\mathfrak{d}$	Nada	R	$\mathfrak{e}$
$\mathfrak{e}$	Nada	P0, R	$\mathfrak{d}$

Fonte: Autor (2024)

Figura 15 – Máquina para o primeiro caso, com $m = 1$, após a computação



Fonte: Autor (2024)

Tabela 6 – Descrição da máquina para o primeiro caso, com $m > 1$

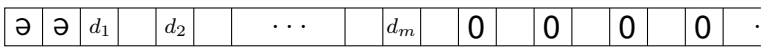
Estado	Símbolo	Operações	Estado final
$\mathfrak{a}$	Nada	Pə, R	$\mathfrak{b}$
$\mathfrak{b}$	Nada	Pə, R	$\mathfrak{c}_1$
$\mathfrak{c}_1$	Nada	Pd_1, R	$\mathfrak{d}_1$
$\mathfrak{d}_1$	Nada	R	$\mathfrak{c}_2$
$\mathfrak{c}_2$	Nada	Pd_2, R	$\mathfrak{d}_2$
$\mathfrak{d}_2$	Nada	R	$\mathfrak{c}_3$

⋮

$\mathfrak{c}_{m-1}$	Nada	Pd_{m-1}, R	$\mathfrak{d}_{m-1}$
$\mathfrak{d}_{m-1}$	Nada	R	$\mathfrak{c}_m$
$\mathfrak{c}_m$	Nada	Pd_m, R	$\mathfrak{d}_m$
$\mathfrak{d}_m$	Nada	R	$\mathfrak{e}$
$\mathfrak{e}$	Nada	P0, R	$\mathfrak{f}$
$\mathfrak{f}$	Nada	R	$\mathfrak{e}$

Fonte: Autor (2024)

Figura 16 – Máquina para o primeiro caso, com $m > 1$, após a computação



Fonte: Autor (2024)

Consegue, portanto, produzir com finitos estados, e usando finitos símbolos, a sequência computada de um número com expansão finita.

ii) Para $m \geq 1$, a situação a ser descrita será semelhante àquela do primeiro caso, mas aplicada ao procedimento guiado a partir da tabela 7.

Partindo, então, do estado $\mathfrak{a}$, após imprimir os primeiros m F-símbolos, o proce-

Tabela 7 – Descrição da máquina para o segundo caso

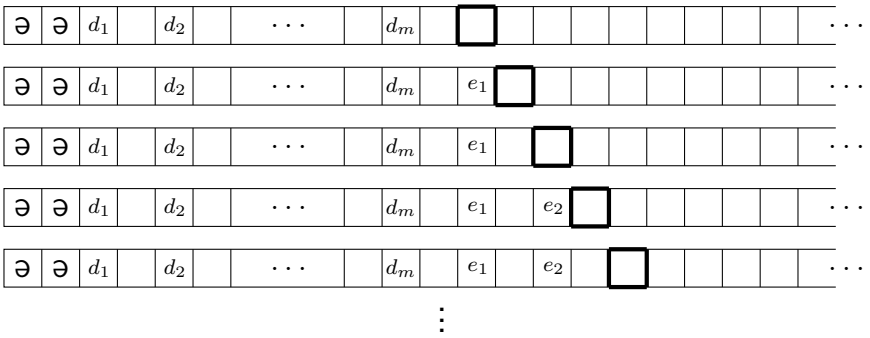
Estado	Símbolo	Operações	Estado final
$\mathfrak{a}$	Nada	Pə, R	$\mathfrak{b}$
$\mathfrak{b}$	Nada	Pə, R	$\mathfrak{c}$
$\mathfrak{c}_1$	Nada	Pd_1, R	$\mathfrak{d}_1$
$\mathfrak{d}_1$	Nada	R	$\mathfrak{c}_2$
$\mathfrak{c}_2$	Nada	Pd_2, R	$\mathfrak{d}_2$
$\mathfrak{d}_2$	Nada	R	$\mathfrak{c}_3$
⋮			
$\mathfrak{c}_m$	Nada	Pd_m, R	$\mathfrak{d}_m$
$\mathfrak{d}_m$	Nada	R	$\mathfrak{e}_1$
$\mathfrak{e}_1$	Nada	Pe_1, R	$\mathfrak{f}_1$
$\mathfrak{f}_1$	Nada	R	$\mathfrak{e}_2$
$\mathfrak{e}_2$	Nada	Pe_2, R	$\mathfrak{f}_2$
$\mathfrak{f}_2$	Nada	R	$\mathfrak{e}_3$
⋮			
$\mathfrak{e}_n$	Nada	Pe_n, R	$\mathfrak{f}_n$
$\mathfrak{f}_n$	Nada	R	$\mathfrak{e}_1$

Fonte: Autor (2024)

dimento da máquina é o que está indicado nas figuras abaixo.

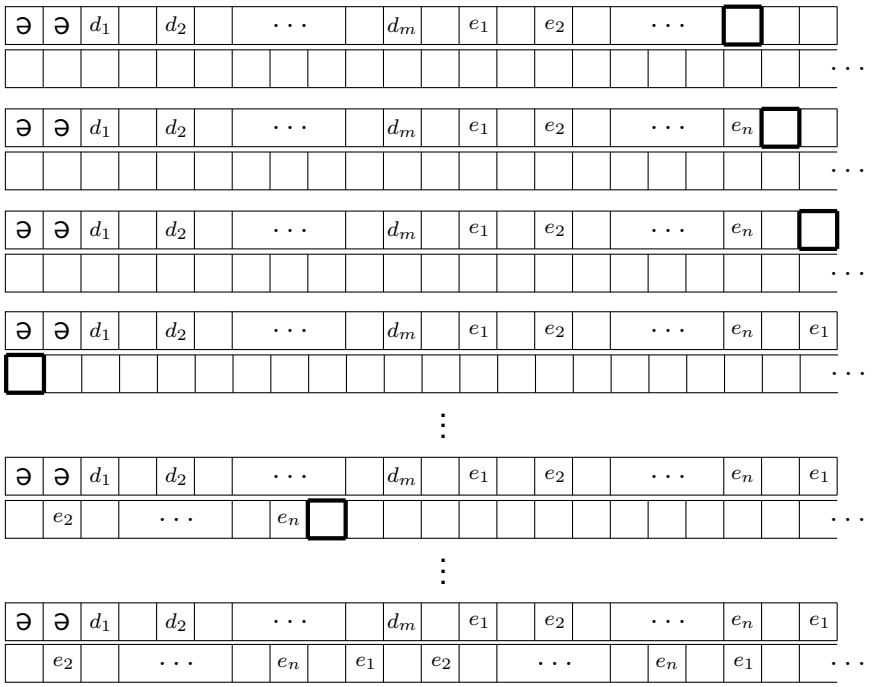
Basicamente, a partir dos dois primeiros estados da primeira coluna da tabela, a máquina de Turing imprimirá os dois ə no começo da computação. A partir dos próximos $2m$ estados, imprimirá cada F-símbolo da parte não periódica como no item **i)**. E a partir dos últimos $2n$ estados, imprimirá cada F-símbolo da parte periódica do número.

Figura 17 – Computação da máquina para o segundo caso (parte 1)



Fonte: Autor (2024)

Figura 18 – Computação da máquina para o segundo caso (parte 2)



Fonte: Autor (2024)

Após percorrer esses $2n$ estados finais, a máquina retornará ao primeiro deles, $\mathfrak{e}_1$, e repetirá este último procedimento com o período indeterminadamente.

Assim, temos uma máquina de Turing que possui apenas $(2m+2n+2)$ estados e 3 símbolos visíveis, e que computa a sequência de nosso número racional. Visto isso, com finitos estados e finitos símbolos produzimos o que almejávamos, e daí temos que todos os números racionais dessa forma são computáveis.

iii) É fácil vermos que o último caso, semelhante ao anterior, pode ser representado por uma máquina com exatamente $(2n + 2)$ estados e 3 símbolos visíveis.

Quanto aos símbolos da máquina de Turing em questão, temos 1 símbolo schwa e 2 símbolos dos algarismos. Uma vez providenciados todos estes elementos, possuímos já o suficiente para a computação desejada. Esses elementos estão em quantidade finita, portanto, os racionais desta forma também provam-se computáveis. ■

Tais números possuem expansão infinita em suas representações, mas com finitos comandos, conseguem ter suas sequências computadas por máquinas de Turing.

Observação 2. Trazemos ainda mais dois resultados, cujas demonstrações, aqui deixaremos de lado.

1. Todos os números algébricos são números computáveis.

2. O número π, que é transcendente, é também um número computável.

Comentando brevemente, é possível demonstrar o primeiro resultado através do método da bissecção. Para um número algébrico qualquer, tomaríamos um polinômio minimal relativo a ele e iniciaríamos com valores da função polinomial, um à sua esquerda e outro à direita, com sinais opostos. O nosso número é raiz dessa função, e aplicando o método, calcularíamos a raiz, pela máquina de Turing, a partir de aproximações sucessivas cada vez mais precisas.

Quanto ao segundo, é possível demonstrar a respectiva computabilidade utilizando a fórmula de Leibniz para π, colocada abaixo.

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}$$

Com ela, calcularíamos aproximações cada vez melhores para π por meio de uma máquina de Turing.

3.3 O CONCEITO DE M-FUNÇÕES

Podemos tomar certos estados da máquina que acabam remetendo a funcionalidades as quais, em um olhar mais global, fazem um trabalho que emula certas ideias humanas. E estas, muitas vezes, aparecem em estruturas semelhantes, onde o que muda apenas é algum símbolo escaneado ou alguma das operações, ou também o próximo estado.

Estados a partir dos quais, por exemplo, é copiado um símbolo para uma extremidade na fita. Ou a partir dos quais são comparados símbolos entre si. A partir dos quais é procurada a primeira aparição de um determinado símbolo até aquele estágio da máquina. Dada a utilidade e versatilidade dos efeitos de estados como esses a ser explorada, é conveniente que possamos ter várias versões semelhantes deles. Versões em que seja possível modificar certo símbolo ou certo estado posterior a ser considerado.

Turing tomou algo para abreviar em uma única representação as versões semelhantes desses estados. Trata-se do conceito de um tipo específico de função. E, antes de a definirmos, apresentaremos outros conceitos.

Primeiramente, vamos encarar definitivamente o preenchimento “em branco” de um quadrado como sendo também um símbolo, ao qual nos referiremos textualmente como $\sqcup$.

Vejamos, agora, um conjunto envolvendo os símbolos da máquina de Turing. Trata-se do chamado **alfabeto da fita**, que consiste no conjunto de todos os símbolos que podem vir a aparecer na máquina, inclusive o ␣. Normalmente é representado pelo símbolo Γ.

Ainda, há o conjunto formado exatamente por todos os estados nos quais a máquina de Turing pode se encontrar. Sem um nome específico, este conjunto é normalmente representado pelo símbolo Q.

Definição 2. Para m_1 e m_2 números naturais, não simultaneamente nulos, chamamos de **m-função** toda função $f : Q^{m_1} \times \Gamma^{m_2} \to Q$.

Quando representamos as imagem das m-funções, os seus argumentos são aqueles m_1 estados e aqueles m_2 símbolos. Pelo modo como foi definida, as imagens da m-função serão sempre estados já presentes na máquina. Assim, podemos ver que ela também age tomando como argumentos elementos que já são imagens de tal função. Voltaremos a falar disso logo.

Enfim, como dissemos, os argumentos com que representamos a imagem de uma m-função serão estados e símbolos. Como veremos, uma m-função, tal como a $\mathfrak{f}$ a ser apresentada a seguir, nos permite utilizar finitos estados na máquina como sendo imagens suas. E a linguagem pela qual ela é inicialmente expressa, com símbolo e estados genéricos como seus argumentos (ou seja, o α, o $\mathfrak{C}$ e o $\mathfrak{B}$) contribui e muito para reduzir o trabalho com que comunicamos a ideia por trás da qual está sua funcionalidade.

É interessante comentar que, na área de trabalho da programação, o que estamos chamando de m-função seria chamado de subrotina, e possuiria este papel de encurtar as ações a serem feitas, levando em conta tais argumentos.

Agora, tratando da presente dissertação, costumamos representar estados genéricos dentro de uma máquina, a serem usados como argumentos, por meio de letras germânicas maiúsculas. E representar símbolos genéricos por meio de letras gregas minúsculas.

Pois bem, apresentaremos, a seguir, certas m-funções e suas descrições. Parte delas será usada no decorrer deste trabalho, e parte delas foi trazida apenas para exemplificar e seguir o roteiro introdutório de Turing.

A m-função *find*

Considere abaixo a m-função que convencionamos chamar de $\mathfrak{f}$ (*find*, "encontrar" em inglês), na qual $\alpha \neq$ ␣.

Atentamos antes que, segundo a linguagem que estamos utilizando, tal como nesta m-função, entendamos o significado do que há na coluna dos símbolos da tabela. Se temos um símbolo β (como, no caso teremos $\mathfrak{d}$ em $\mathfrak{f}$), então "Não β" quer

dizer todos os outros símbolos possíveis com que o quadrado escaneado estaria em seu interior, incluindo o caso de ele estar em branco, e excluindo apenas o caso de estar com β.

Tabela 8 – Descrição a partir da m-função *find*

Estado	Símbolo	Operações	Estado final
$\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$	ə	L	$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$
	Não ə	L	$\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$
$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$	α		$\mathfrak{C}$
	Não α e não ␣	R	$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$
	␣	R	$\mathfrak{f}_2(\mathfrak{C}, \mathfrak{B}, \alpha)$
$\mathfrak{f}_2(\mathfrak{C}, \mathfrak{B}, \alpha)$	α		$\mathfrak{C}$
	Não α e não ␣	R	$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$
	␣	R	$\mathfrak{B}$

Fonte: Turing (1937)

Esta m-função nos leva a um estado que age a partir de três argumentos: os estados $\mathfrak{C}$ e $\mathfrak{B}$, e o símbolo α. Neste caso, $\mathfrak{f}$ é uma função de domínio $Q^2 \times \Gamma$. Ela mostra consigo as m-funções $\mathfrak{f}_1$ e $\mathfrak{f}_2$, ambas também de domínio $Q^2 \times \Gamma$.

A máquina, a partir do estado $\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$, age para encontrar o símbolo α mais à esquerda na fita, e vai para o estado $\mathfrak{C}$. Se não há nenhum exemplar de α, a máquina vai para o estado $\mathfrak{B}$.

Dentro de uma máquina de Turing, as m-funções com seus argumentos significam o seguinte. Utilizando o caso desta para ilustrarmos, digamos que sejam $\Gamma = \{\alpha_0, \alpha_1, \ldots, \alpha_{m-1}\}$ o seu alfabeto da fita, e $Q = \{\mathfrak{q}_0, \mathfrak{q}_1, \ldots, \mathfrak{q}_{n-1}\}$. Ocorre que um certo número de estados da máquina é exatamente da forma $\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$ enquanto o restante dos estados não o é.

Por exemplo, em um potencial momento, a máquina é capaz de passar para o estado correspondente a $\mathfrak{f}(\mathfrak{q}_4, \mathfrak{q}_2, \alpha_7)$, ao qual a ação subsequente é descrita na tabela. A partir deste estado, ela procura pelo símbolo α_7 mais à esquerda na fita, muda para o estado $\mathfrak{q}_4$ quando o encontrar e muda para $\mathfrak{q}_2$ se não o encontrar. Podemos programar de modo a tomar quantos e quais símbolos forem quistos e os combinar com os estados que também o forem.

Sabemos que a máquina tem finitos estados. E pode acontecer de existirem estados que são imagens desta m-função $\mathfrak{f}$, nos quais algum dos seus argumentos é também uma imagem de $\mathfrak{f}$. E ainda pode ocorrer de este argumento conter, como um de seus argumentos, outro estado que seja uma imagem de $\mathfrak{f}$. E isso prosseguir um certo número de vezes, ou até mesmo potenciais infinitas vezes. Mas, por sabermos que há finitos estados, isto ir ao infinito dentro de nossa máquina de Turing não se apresenta como um problema, na questão do conceito.

Pois, digamos, os estados $\mathfrak{q}_{k_1}$, $\mathfrak{q}_{k_2}$, ..., $\mathfrak{q}_{k_l}$ sejam todas as imagens de $\mathfrak{f}$. Então,

uma situação com maior abrangência em que isso poderia ocorrer se daria da seguinte maneira. Tomamos algum desses estados como argumento de $\mathfrak{f}$, e a imagem nos dá um segundo estado entre eles, distinto do anterior. Tomamos esse segundo como argumento, e a imagem é um terceiro estado distinto dos dois anteriores. E assim vamos, até estarmos na imagem correspondente a um $(l-1)$-ésimo estado distinto dos anteriores. E tomando este como argumento, a próxima imagem será o l-ésimo e último estado distinguível. Após este ser tomado como argumento, as próximas imagens de $\mathfrak{f}$ só poderão ser iguais a algum dos estados já citados. Não haverá novidade de estados.

E ainda há a questão da funcionalidade da máquina. Pois mesmo que não haja impedimento na definição para tais recursões, poderíamos pensar no problema de as recursões infinitas "prenderem" a máquina no mesmo tipo de ação, e ela não pudesse prosseguir preenchendo os F-quadrados da fita como deveria.

No caso de $\mathfrak{f}$, pensemos no estado $\mathfrak{q}^*$, a partir do qual a máquina procura pelo símbolo α_5 e, se o encontrar ou não, muda para o estado $\mathfrak{q}^{**}$, a partir do qual procura por α_6 e, se o encontrar ou não, muda para o estado $\mathfrak{q}^*$ novamente. Ela permanecerá nesse ciclo, sem ter fim, caracterizando-se como uma máquina circular.

No entanto, sabendo de situações assim, ao ser programada, a máquina já deve ser planejada de modo que só fará recursões dentro da m-função $\mathfrak{f}$ enquanto lhe forem convenientes para imprimir a sequência respectiva. Prevendo a ordem na qual ela passará pelos símbolos escaneados e estados, quem fosse responsável por definir o estado inicial dessa máquina e suas transições já os deve pensar de modo a evitar esses casos de repetição.

Pois, mesmo que a m-função esteja definida para todos os estados e símbolos no domínio, há aqueles estados e símbolos com os quais ela nunca terá contato de fato durante a ação da máquina. Esse seria o caso de α_5 com $\mathfrak{q}^*$ e de α_6 com $\mathfrak{q}^{**}$ em $\mathfrak{f}$, no sentido de evitar que a máquina caia nesses estados. Esclarecido isso, podemos continuar utilizando o conceito das m-funções devidamente.

As m-funções *erase*

Agora, apresentamos a m-função que chamaremos de $\mathfrak{e}$, sendo esta de três argumentos. Recebe tal nome devido à palavra *erase*, de "apagar", em inglês.

Tabela 9 – Descrição a partir da m-função *erase* de 3 argumentos

Estado	Símbolo	Operações	Estado final
$\mathfrak{e}(\mathfrak{C},\mathfrak{B},\alpha)$	Qualquer		$\mathfrak{f}(\mathfrak{e}_1(\mathfrak{C},\mathfrak{B},\alpha),\mathfrak{B},\alpha)$
$\mathfrak{e}_1(\mathfrak{C},\mathfrak{B},\alpha)$	Qualquer	E	$\mathfrak{C}$

Fonte: Turing (1937)

Deixamos claro que, a partir de agora, quando indicarmos "Qualquer" na coluna

dos símbolos escaneados, estamos nos referindo a todo símbolo possível, incluindo a situação em que o quadrado escaneado esteja em branco.

Aqui temos a função $\mathfrak{e} : Q^2 \times \Gamma \to Q$. Mostra consigo a função $\mathfrak{e}_1$, de mesmo domínio. A máquina, a partir do estado $\mathfrak{e}(\mathfrak{C}, \mathfrak{B}, \alpha)$, age para encontrar e apagar o primeiro símbolo α da fita, isto é, o α mais à esquerda. E então muda para o estado $\mathfrak{C}$. Se não há outro exemplar do símbolo, muda para o estado $\mathfrak{B}$.

Com esta m-função, apresentamos a m-função $\mathfrak{e}$ de dois argumentos.

Tabela 10 – Descrição a partir da m-função *erase* de 2 argumentos

Estado	**Símbolo**	**Operações**	**Estado final**
$\mathfrak{e}(\mathfrak{B}, \alpha)$	Qualquer		$\mathfrak{e}(\mathfrak{e}(\mathfrak{B}, \alpha), \mathfrak{B}, \alpha)$

Fonte: Turing (1937)

Neste caso, temos a função $\mathfrak{e} : Q \times \Gamma \to Q$.

A partir do estado $\mathfrak{e}(\mathfrak{B}, \alpha)$, a máquina apaga todos os símbolos α, e então muda para o estado $\mathfrak{B}$. Como, durante a atividade da máquina, há com certeza uma quantidade finita de exemplares de um mesmo símbolo na fita, podemos não nos preocupar se a máquina ficará presa em consecutivas vezes repetindo o estado $\mathfrak{e}$ de dois argumentos. Alguma hora ela terá apagado todos os símbolos α e então mudará para o estado $\mathfrak{B}$, escolhido convenientemente.

A m-função *print at the end*

Apresentamos a m-função $\mathfrak{pe}$. Significa *print at the end*, “imprima no final”.

Tabela 11 – Descrição a partir da m-função *print at the end*

Estado	**Símbolo**	**Operações**	**Estado final**
$\mathfrak{pe}(\mathfrak{C}, \beta)$	Qualquer		$\mathfrak{f}(\mathfrak{pe}_1(\mathfrak{C}, \beta), \mathfrak{C}, \mathfrak{ə})$
$\mathfrak{pe}_1(\mathfrak{C}, \beta)$	Não $\sqcup$	R, R	$\mathfrak{pe}_1(\mathfrak{C}, \beta)$
	$\sqcup$	Pβ	$\mathfrak{C}$

Fonte: Turing (1937)

Já que consideramos este “*em branco*” como símbolo, passível de impressão, então as operações P$_\sqcup$ e E são equivalentes.

Mesmo assim, quando nós estivermos utilizando expressões como “o fim da sequência de símbolos na fita” durante as transições da máquina, esclarecemos que estamos considerando como os símbolos em questão, todos, exceto o símbolo *em branco*. Afinal, a princípio, a máquina já estaria com infinitos quadrados em sequência preenchidos com este símbolo. E claramente nos estamos referindo aos quadrados com símbolos “visíveis” ao longo da fita, e não à infinitude dos outros onde se inicia uma sequência toda de espaços vazios onde a cabeça não deixou nada novo.

Voltemos a falar da m-função $\mathfrak{pe}$ em específico. Temos $\mathfrak{pe} : Q \times \Gamma \to Q$, que mostra consigo a função $\mathfrak{pe}_1 : Q \times \Gamma \to Q$. A partir do estado $\mathfrak{pe}(\mathfrak{C}, \beta)$, a máquina imprime o símbolo β no fim da sequência de símbolos da fita e então muda para o estado $\mathfrak{C}$.

É claro, para que o estado faça jus ao nome, a máquina deve ser programada para que, ao longo de suas transições, os F-quadrados preenchidos não fiquem a uma distância de quatro unidades do F-quadrado mais próximo à esquerda. Se não, no estado em questão, a cabeça chegaria a este F-quadrado em branco entre dois F-quadrados preenchidos, e imprimiria β como se ali fosse o fim atual da sequência. Para isso é necessária atenção ao programar.

As m-funções *left* e *right*

Vejamos agora as m-funções $\mathfrak{l}$ e $\mathfrak{r}$. Significam, respectivamente, *left* e *right*, isto é, esquerda e direita.

Tabela 12 – Descrição a partir das m-funções *left* e *right*

Estado	Símbolo	Operações	Estado final
$\mathfrak{l}(\mathfrak{C})$	Qualquer	L	$\mathfrak{C}$
$\mathfrak{r}(\mathfrak{C})$	Qualquer	R	$\mathfrak{C}$
$\mathfrak{f}'(\mathfrak{C}, \mathfrak{B}, \alpha)$	Qualquer		$\mathfrak{f}(\mathfrak{l}(\mathfrak{C}), \mathfrak{B}, \alpha)$
$\mathfrak{f}''(\mathfrak{C}, \mathfrak{B}, \alpha)$	Qualquer		$\mathfrak{f}(\mathfrak{r}(\mathfrak{C}), \mathfrak{B}, \alpha)$

Fonte: Turing (1937)

Ambas são funções de domínio Q. O trabalho da máquina a partir dos estados respectivos é simples: deslocar a cabeça para a esquerda ou para a direita, e então mudar para o estado $\mathfrak{C}$. Como podemos ver, aqui também são mostradas as m-funções de nosso interesse $\mathfrak{f}'$ e $\mathfrak{f}''$, ambas de domínio $Q^2 \times \Gamma$.

A partir do estado $\mathfrak{f}'(\mathfrak{C}, \mathfrak{B}, \alpha)$, a máquina faz o mesmo que faria com $\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$, mas ainda se move uma unidade para a esquerda após encontrar o símbolo respectivo logo antes de mudar para o estado $\mathfrak{C}$. O estado $\mathfrak{f}''(\mathfrak{C}, \mathfrak{B}, \alpha)$ provoca o mesmo, só que para o sentido da direita.

A partir daqui, deixaremos de sempre escrever o domínio e contradomínio que definem as m-funções, pois os exemplares anteriores já dão conta de mostrar como as devemos entender.

A m-função *copy*

Apresentamos, agora, a m-função $\mathfrak{c}$, que significa *copy*, de “copiar”.

A partir do estado $\mathfrak{c}(\mathfrak{C}, \mathfrak{B}, \alpha)$, a máquina imprime, no fim da sequência de símbolos da fita, o primeiro símbolo marcado por α, e então ela muda para o estado $\mathfrak{C}$.

Tabela 13 – Descrição a partir da m-função *copy*

Estado	Símbolo	Operações	Estado final
$\mathfrak{c}(\mathfrak{C}, \mathfrak{B}, \alpha)$	Qualquer		$\mathfrak{f}'(\mathfrak{c}_1(\mathfrak{C}), \mathfrak{B}, \alpha)$
$\mathfrak{c}_1(\mathfrak{C})$	β		$\mathfrak{pe}(\mathfrak{C}, \beta)$

Fonte: Turing (1937)

A sintaxe do estado $\mathfrak{c}_1$, pelo destaque ao símbolo escaneado, se mostra para nós pouco habitual. Ocorre que, de acordo com o símbolo escaneado, a máquina torna ele próprio, seja qual for, o segundo argumento do próximo estado, que é baseado na m-função $\mathfrak{pe}$.

Tenha a nossa máquina de Turing um total de m^* símbolos distintos que poderão ser escaneados durante o estado $\mathfrak{c}_1$. Sejam $\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_{m^*}}$ esses símbolos. Então temos programada na máquina, na realidade, a seguinte ação a partir do referido estado.

Tabela 14 – Explicação da tabela para a m-função $\mathfrak{c}_1$

Estado	Símbolo	Operações	Estado final
$\mathfrak{c}_1$	α_{i_1}		$\mathfrak{pe}\,(\mathfrak{C}, \alpha_{i_1})$
	α_{i_2}		$\mathfrak{pe}\,(\mathfrak{C}, \alpha_{i_2})$
	$\vdots$	$\vdots$	$\vdots$
	$\alpha_{i_{m^*}}$		$\mathfrak{pe}\big(\mathfrak{C}, \alpha_{i_{m^*}}\big)$

Fonte: Petzold (2008)

E a lógica deste exemplo será a mesma lógica de outros estados que adotarem o símbolo escaneado como argumento do estado seguinte.

A m-função *copy and erase*

Apresentamos agora a m-função $\mathfrak{ce}$, de três e de dois argumentos, que significa *copy and erase*, isto é, "copie e apague".

Tabela 15 – Descrição a partir da m-função *copy and erase*

Estado	Símbolo	Operações	Estado final
$\mathfrak{ce}(\mathfrak{C}, \mathfrak{B}, \alpha)$	Qualquer		$\mathfrak{c}(\mathfrak{e}(\mathfrak{C}, \mathfrak{B}, \alpha), \mathfrak{B}, \alpha)$
$\mathfrak{ce}(\mathfrak{B}, \alpha)$	Qualquer		$\mathfrak{ce}(\mathfrak{ce}(\mathfrak{B}, \alpha), \mathfrak{B}, \alpha)$

Fonte: Turing (1937)

A partir do estado $\mathfrak{ce}(\mathfrak{B}, \alpha)$, a máquina copia, em ordem, até o fim da fita, todos os símbolos marcados por α e apaga os marcadores α no processo. E então muda para o estado $\mathfrak{B}$.

Isso funciona devido a termos colocado o próprio estado $\mathfrak{ce}(\mathfrak{B}, \alpha)$ como primeiro argumento na m-função correspondente ao estado sucessor. O que estamos

indicando é que, após copiar o primeiro símbolo marcado por α para o fim da fita e apagar o marcador, a máquina retornará para o estado $\mathfrak{ce}(\mathfrak{B}, \alpha)$ e fará a mesma coisa. Isso acaba quando não houver mais símbolos marcados por α.

O fato de $\mathfrak{ce}(\mathfrak{B}, \alpha)$ ser argumento no estado sucessor de si mesmo não causa problema de definição circular. Afinal, o que a tabela traz são as consequências da leitura da máquina sobre os símbolos durante o estado $\mathfrak{ce}(\mathfrak{B}, \alpha)$. Acontece que $\mathfrak{ce}(\mathfrak{B}, \alpha)$ já é, por si só, um estado da máquina. As operações e sucessões que dele seguem poderiam ser outras. O que a tabela define são as transições da máquina a partir de tal estado.

Para o entendermos melhor, chamemos o estado $\mathfrak{ce}(\mathfrak{ce}(\mathfrak{B}, \alpha), \mathfrak{B}, \alpha)$ de $\mathfrak{q}^*$. Digamos que, com a cabeça sobre um símbolo β, a máquina esteja sobre o estado $\mathfrak{ce}(\mathfrak{B}, \alpha)$. Não há operações imediatas e o estado sucessor será então $\mathfrak{q}^*$. A partir dele ocorrerão as operações e sucessões respectivas, e só em um futuro relativamente próximo, se houver símbolo marcado por α, é que retornaremos ao estado $\mathfrak{ce}(\mathfrak{B}, \alpha)$. Seguiremos fazendo o que foi descrito. A máquina funciona normalmente, sem impedimento algum.

As m-funções *replace*

Assumamos, para este momento, que α e β sejam marcadores. Então apresentamos a m-função $\mathfrak{re}$, de quatro argumentos, que significa *replace*, isto é, "substituir".

Tabela 16 – Descrição a partir da m-função *replace* de 4 argumentos

Estado	Símbolo	Op.	Estado final
$\mathfrak{re}(\mathfrak{C}, \mathfrak{B}, \alpha, \beta)$	Qualquer		$\mathfrak{f}(\mathfrak{re}_1(\mathfrak{C}, \mathfrak{B}, \alpha, \beta), \mathfrak{B}, \alpha)$
$\mathfrak{re}_1(\mathfrak{C}, \mathfrak{B}, \alpha, \beta)$	Qualquer	E, Pβ	$\mathfrak{C}$

Fonte: Turing (1937)

Neste caso, temos a função $\mathfrak{re} : Q^2 \times \Gamma^2 \to Q$. E ela mostra consigo a função $\mathfrak{re}_1 : Q^2 \times \Gamma^2 \to Q$.

A partir do estado $\mathfrak{re}(\mathfrak{C}, \mathfrak{B}, \alpha, \beta)$, a máquina substitui o primeiro símbolo α da fita pelo símbolo β, e então muda para o estado $\mathfrak{C}$. Mas se não houver nenhum α na fita, então ela muda para o estado $\mathfrak{B}$.

E com isso, apresentamos outra m-função $\mathfrak{re}$, mas que desta vez possui três argumentos.

Tabela 17 – Descrição a partir da m-função *replace* de 3 argumentos

Estado	Símbolo	Operações	Estado final
$\mathfrak{re}(\mathfrak{B}, \alpha, \beta)$	Qualquer		$\mathfrak{re}(\mathfrak{re}(\mathfrak{B}, \alpha, \beta), \mathfrak{B}, \alpha, \beta)$

Fonte: Turing (1937)

A partir do estado $\mathfrak{re}(\mathfrak{B}, \alpha, \beta)$, a máquina substitui todos os símbolos α da fita por β, desde o primeiro até o último, e então muda para o estado $\mathfrak{B}$.

Como no caso da função $\mathfrak{ce}$ de dois argumentos, aqui não há problema de definição circular. A explicação para tal é similar.

A m-função *copy and replace*

Apresentamos a m-função $\mathfrak{cr}$, de três e de dois argumentos, que significa *copy and replace*, isto é, "copie e substitua".

Tabela 18 – Descrição a partir da m-função *copy and replace*

Estado	Símbolo	Op.	Estado final
$\mathfrak{cr}(\mathfrak{C}, \mathfrak{B}, \alpha)$	Qualquer		$\mathfrak{c}(\mathfrak{re}(\mathfrak{C}, \mathfrak{B}, \alpha, \alpha), \mathfrak{B}, \alpha)$
$\mathfrak{cr}_1(\mathfrak{B}, \alpha)$	Qualquer		$\mathfrak{cr}(\mathfrak{cr}(\mathfrak{B}, \alpha), \mathfrak{re}(\mathfrak{B}, \alpha, \alpha), \alpha)$

Fonte: Turing (1937)

A partir do estado $\mathfrak{cr}(\mathfrak{B}, \alpha)$, a máquina cumpre o mesmo papel que teria a partir do estado $\mathfrak{ce}(\mathfrak{B}, \alpha)$, com a diferença de que os marcadores α não são apagados ao final do procedimento.

A m-função *compare*

Apresentamos, neste momento, a m-função $\mathfrak{cp}$, que significa *compare*, isto é, comparar.

Ao contemplarmos sua imagem, notemos a sutil diferença entre o símbolo do primeiro e do terceiro argumento. São, respectivamente, as letras C e E germânicas.

Tabela 19 – Descrição a partir da m-função *compare*

Estado	Símbolo	Op.	Estado final
$\mathfrak{cp}(\mathfrak{C}, \mathfrak{D}, \mathfrak{E}, \alpha, \beta)$	Qualquer		$\mathfrak{f}'(\mathfrak{cp}_1(\mathfrak{C}, \mathfrak{D}, \beta), \mathfrak{f}(\mathfrak{D}, \mathfrak{E}, \beta), \alpha)$
$\mathfrak{cp}_1(\mathfrak{C}, \mathfrak{D}, \beta)$	γ		$\mathfrak{f}'(\mathfrak{cp}_2(\mathfrak{C}, \mathfrak{D}, \gamma), \mathfrak{D}, \beta)$
$\mathfrak{cp}_2(\mathfrak{C}, \mathfrak{D}, \gamma)$	γ		$\mathfrak{C}$
	Não γ		$\mathfrak{D}$

Fonte: Turing (1937)

Ocorrerá que o primeiro símbolo marcado com α é comparado ao primeiro símbolo marcado com β. A partir do estado $\mathfrak{cp}(\mathfrak{C}, \mathfrak{D}, \mathfrak{E}, \alpha, \beta)$, se não há α e nem β, então a máquina muda para o estado $\mathfrak{E}$; se há α e β e os símbolos marcados com eles são iguais, então ela muda para o estado $\mathfrak{C}$; de outro modo, muda para $\mathfrak{D}$.

As m-funções *compare and erase*

Tabela 20 – Descrição a partir da m-função *compare and erase* de 5 argumentos

Estado	Símbolo	Op.	Estado final
$\mathfrak{cpe}(\mathfrak{C}, \mathfrak{D}, \mathfrak{E}, \alpha, \beta)$	Qualquer		$\mathfrak{cp}(\mathfrak{e}(\mathfrak{e}(\mathfrak{C}, \mathfrak{C}, \alpha)\mathfrak{C}, \alpha), \mathfrak{D}, \mathfrak{E}, \alpha, \beta)$

Fonte: Turing (1937)

Apresentamos agora a m-função $\mathfrak{cpe}$, de cinco argumentos, que significa *compare and erase*, isto é, "compare e apague".

A partir do estado $\mathfrak{cpe}(\mathfrak{C}, \mathfrak{D}, \mathfrak{E}, \alpha, \beta)$, a máquina cumpre o mesmo papel que teria com o estado $\mathfrak{cp}(\mathfrak{C}, \mathfrak{D}, \mathfrak{E}, \alpha, \beta)$, com a diferença de que, quando os símbolos marcados pelo primeiro α e pelo primeiro β forem iguais, os primeiros marcadores α e β serão apagados.

Com isso, apresentamos a m-função $\mathfrak{cpe}$ de quatro argumentos.

Tabela 21 – Descrição a partir da m-função *compare and erase* de 4 argumentos

Estado	Símbolo	Op.	Estado final
$\mathfrak{cpe}(\mathfrak{D}, \mathfrak{E}, \alpha, \beta)$	Qualquer		$\mathfrak{cpe}(\mathfrak{cpe}(\mathfrak{D}, \mathfrak{E}, \alpha, \beta), \mathfrak{D}, \mathfrak{E}, \alpha, \beta)$

Fonte: Turing (1937)

A partir do estado $\mathfrak{cpe}(\mathfrak{D}, \mathfrak{E}, \alpha, \beta)$, a máquina prossegue o que faria com o estado $\mathfrak{cpe}(\mathfrak{C}, \mathfrak{D}, \mathfrak{E}, \alpha, \beta)$, pois agora, se os primeiros símbolos marcados com α e β são iguais, ela apaga esses marcadores e compara os próximos símbolos marcados por eles. E assim segue, em ordem, até que não encontre mais um par de símbolos marcados por α e β onde ambos sejam iguais. Daí ela vai, ao final, mudar, ou para o estado $\mathfrak{E}$, ou para o $\mathfrak{D}$, como na versão de cinco argumentos.

As m-funções *g*

Apresentamos agora a m-função $\mathfrak{g}$, de um e de dois argumentos.

Tabela 22 – Descrição a partir da m-função $\mathfrak{g}$

Estado		Símbolo	Operações	Estado final
$\mathfrak{g}(\mathfrak{C})$	$\left\{\right.$	Não $\sqcup$	R	$\mathfrak{g}(\mathfrak{C})$
		$\sqcup$	R	$\mathfrak{g}_1(\mathfrak{C})$
$\mathfrak{g}_1(\mathfrak{C})$	$\left\{\right.$	Não $\sqcup$	R	$\mathfrak{g}(\mathfrak{C})$
		$\sqcup$		$\mathfrak{C}$
$\mathfrak{g}(\mathfrak{C}, \alpha)$		Qualquer		$\mathfrak{g}(\mathfrak{g}_1(\mathfrak{C}, \alpha))$
$\mathfrak{g}_1(\mathfrak{C}, \alpha)$	$\left\{\right.$	α		$\mathfrak{C}$
		Não α	L	$\mathfrak{g}_1(\mathfrak{C}, \alpha)$

Fonte: Turing (1937)

Enquanto na m-função $\mathfrak{f}$ a cabeça anda em direção ao início da fita e encontra um dado símbolo, nas m-funções $\mathfrak{g}$ o trabalho se dá indo até o fim da fita. Para isso, a

partir do estado $\mathfrak{g}(\mathfrak{C})$, a cabeça da máquina se move no sentido da direita até encontrar dois quadrados em branco seguidos. E então muda para o estado $\mathfrak{C}$.

E a partir do estado $\mathfrak{g}(\mathfrak{C}, \alpha)$, a cabeça encontra o último símbolo igual a α, e então muda para o estado $\mathfrak{C}$.

Expandindo argumentos de algumas m-funções

Apresentamos agora, baseadas na m-função $\mathfrak{ce}$ (*copy and erase*) de dois argumentos, as m-funções $\mathfrak{ce}_2$ e $\mathfrak{ce}_3$.

Tabela 23 – Descrição a partir da m-função *copy and erase* de mais argumentos

Estado	Símbolo	Operações	Estado final
$\mathfrak{ce}_2(\mathfrak{B}, \alpha, \beta)$	Qualquer		$\mathfrak{ce}(\mathfrak{ce}(\mathfrak{B}, \beta), \alpha)$
$\mathfrak{ce}_3(\mathfrak{B}, \alpha, \beta, \gamma)$	Qualquer		$\mathfrak{ce}(\mathfrak{ce}_2(\mathfrak{B}, \beta, \gamma), \alpha)$

Fonte: Turing (1937)

Lembremos que, a partir do estado $\mathfrak{ce}(\mathfrak{B}, \alpha)$, a máquina copiava, em ordem, todos os símbolos marcados com α para o fim da fita, apagava tais marcadores, e mudava para $\mathfrak{B}$. Pois que, a partir do estado $\mathfrak{ce}_2(\mathfrak{B}, \alpha, \beta)$, a máquina copia os marcados por α igualmente, e apaga estes marcadores. Mas depois copia os marcados por β, e apaga estes marcadores, antes de mudar para $\mathfrak{B}$.

Similarmente, a partir do estado $\mathfrak{ce}_3(\mathfrak{B}, \alpha, \beta, \gamma)$, a máquina faz, tomando α e β, o mesmo que faria a partir de $\mathfrak{ce}_2(\mathfrak{B}, \alpha, \beta)$, mas ainda toma o terceiro marcador, γ. E então age igualmente, copiando ao fim da máquina, em ordem, os símbolos marcados por ele, apagando tal marcador no processo. E então muda para $\mathfrak{B}$.

E assim poderíamos continuar fazendo adicionando um quarto marcador, utilizando o mesmo raciocínio. Este exemplo nos ajuda a notar como, em certas ocasiões, é possível criarmos novas m-funções para estender a ação que a máquina tomaria com uma m-função original, mas adicionando novos argumentos, e com uma linguagem relativamente simples.

E para vermos esse mesmo princípio novamente, apresentamos agora a m-função $\mathfrak{pe}_2$, cuja ação fornecida é baseada na ação fornecida por $\mathfrak{pe}$ (*print at the end*).

Tabela 24 – Descrição a partir da m-função *print at the end* de mais argumentos

Estado	Símbolo	Operações	Estado final
$\mathfrak{pe}_2(\mathfrak{C}, \alpha, \beta)$	Qualquer		$\mathfrak{pe}(\mathfrak{pe}(\mathfrak{C}, \beta), \alpha)$

Fonte: Turing (1937)

Recordando, a partir do estado $\mathfrak{pe}(\mathfrak{C}, \alpha)$, a máquina imprimia o símbolo α no F-quadrado em branco ao fim da fita, e então mudava para o estado $\mathfrak{C}$. E, a partir do estado $\mathfrak{pe}_2(\mathfrak{C}, \alpha, \beta)$, a máquina imprime, respectivamente, os símbolos α e β nos dois últimos F-quadrados em branco ao fim da fita, e então muda para o estado $\mathfrak{C}$.

E, novamente, podemos estender esse comportamento para um maior número de argumentos com símbolos. Para o número de três destes argumentos, teríamos o estado $\mathfrak{pe}_3(\mathfrak{C}, \alpha, \beta, \gamma)$, cujo estado sucessor seria $\mathfrak{pe}(\mathfrak{pe}_2(\mathfrak{C}, \beta, \gamma), \alpha)$. Para quatro, ou cinco, ou mais de tais argumentos, a mesma lógica seria seguida.

E as ações da máquina a partir destes estados são facilmente dedutíveis. Imprimirá todos os símbolos dos argumentos, em ordem, na mesma quantidade de F-quadrados em branco ao fim da fita, e mudará para o estado $\mathfrak{C}$ logo após isso.

A m-função *erase* de 1 argumento

E apresentamos a m-função $\mathfrak{e}$ de um argumento.

Tabela 25 – Descrição a partir da m-função *erase* de 1 argumento

Estado	Símbolo	Operações	Estado final
$\mathfrak{e}(\mathfrak{C})$	ə	R	$\mathfrak{e}_1(\mathfrak{C})$
	Não ə e não ␣	L	$\mathfrak{e}(\mathfrak{C})$
	Não ␣	L	$\mathfrak{e}(\mathfrak{C})$
$\mathfrak{e}_1(\mathfrak{C})$	Não ␣	R, E, R	$\mathfrak{e}_1(\mathfrak{C})$
	␣		$\mathfrak{C}$

Fonte: Turing (1937)

A partir do estado $\mathfrak{e}(\mathfrak{C})$, a máquina apaga desde o primeiro até o último marcador, e então, quando a cabeça chega ao fim da fita, a máquina muda para o estado $\mathfrak{C}$.

Notemos ainda que, por vezes, tais m-funções necessitam de alguma m-função auxiliar seguinte, cujos estados que são suas imagens podem provocar o escaneamento de mais outro símbolo, fazer mais alguma operação (como o são $\mathfrak{f}$ com $\mathfrak{f}_1$e $\mathfrak{f}_2$, ou $\mathfrak{pe}$ com $\mathfrak{pe}_1$). Mas cumprem seus papéis sem precisarem de infinitas especificações.

3.4 CONVENÇÕES DE TURING E DE SIPSER

Observação 3. Separamos aqui dois conjuntos que constantemente usaremos nesta seção.

O conjunto $\Gamma = \{\alpha_0, \alpha_1, \ldots, \alpha_{m-1}\}$, a ser usado como alfabeto da fita de uma máquina de Turing.

E o conjunto $Q = \{\mathfrak{q}_0, \mathfrak{q}_1, \ldots, \mathfrak{q}_{n-1}\}$, a ser usado como o conjunto de estados possíveis de uma máquina de Turing.

Turing, quando trabalha com certo rigor a partir dos estados de suas máquinas, acaba nos levando, em dado momento, à conclusão de que o total de números computáveis existe em cardinalidade enumerável. Mais à frente demonstraremos essa

afirmação. Mas por enquanto, veremos que, até chegar lá, Turing teve de padronizar as ações decorrentes dos estados.

Para tal, ele adotou determinadas convenções. Enquanto íamos introduzindo a natureza e o trabalho das máquinas de Turing, utilizamos, nesta dissertação, essas mesmas convenções. A primeira vez que trouxemos um estado sobre o qual elas não eram obedecidas foi ao apresentarmos a m-função $\mathfrak{pe}$, *print at the end*. Mas afinal, quais são exatamente essas convenções?

Ele definiu que um estado convencional causaria imediatamente no mínimo nenhuma, e no máximo duas operações. Por isso, aqui, decidimos chamar esse tipo de estados de estado "passo único".

Se sob tal estado a máquina tem uma única operação, ela seria, ou imprimir um símbolo (visível) no quadrado escaneado, ou apagar o símbolo do quadrado escaneado, ou mover a cabeça uma unidade para a direita, ou a mover uma unidade para a esquerda. Se duas, a primeira seria, ou a impressão de um símbolo, ou o apagamento do símbolo escaneado; e a segunda seria, ou mover a cabeça uma unidade para a direita, ou a mover uma unidade para a esquerda.

É disso que se tratam as convenções. Até apresentarmos a m-função $\mathfrak{pe}$, as havíamos seguido, sem que anunciássemos sempre. Mas agora, olhemos para a tabela do esquema em torno de $\mathfrak{pe}$, mais especificamente no que diz respeito ao estado $\mathfrak{pe}_1(\mathfrak{C}, \beta)$. Tanto a partir deste estado como de tantos outros que Turing apresenta, normalmente imagens de m-funções, quebram-se essas regras.

Recordando, abaixo temos a tabela onde $\mathfrak{pe}_1(\mathfrak{C}, \beta)$ foi apresentado.

Tabela 26 – Recapitulação da descrição a partir de *print at the end*

Estado		**Símbolo**	**Operações**	**Estado final**
$\mathfrak{pe}(\mathfrak{C}, \beta)$		Qualquer		$\mathfrak{f}(\mathfrak{pe}_1(\mathfrak{C}, \beta), \mathfrak{C}, \mathfrak{d})$
$\mathfrak{pe}_1(\mathfrak{C}, \beta)$	$\left\{\right.$	Não $\sqcup$	R, R	$\mathfrak{pe}_1(\mathfrak{C}, \beta)$
		$\sqcup$	Pβ	$\mathfrak{C}$

Fonte: Turing (1937)

No caso, a partir de $\mathfrak{pe}_1(\mathfrak{C}, \beta)$, temos, como operação da máquina em um único estado, o movimento da cabeça em duas unidades à direita. E Turing de fato se utiliza dessa m-função em algum momento para computar, como também de outras que fogem ao padrão que ele mesmo estabeleceu.

Mas isso não é um grande problema. Se Turing quer que os estados de suas máquinas obedeçam as suas convenções, basta aqui, tal como em outras situações, entender o presente estado "multipasso" como uma composição de estados "passo único".

Neste caso de $\mathfrak{pe}_1(\mathfrak{C}, \beta)$, imaginemos que, na situação onde a cabeça escaneou um símbolo que não $\sqcup$, a máquina se encontra em um estado $\mathfrak{q}^*$, que é imagem de

uma m-função com os argumentos $\mathfrak{C}$ e β, a partir do qual ela anda uma unidade para a direita. Daí, ela muda para um estado $\mathfrak{q}^{**}$, também imagem de uma m-função com os mesmos argumentos, tal que, para qualquer que seja o símbolo escaneado, anda uma unidade para a direita, e então muda de volta para $\mathfrak{q}^{*}$.

Desta maneira, conseguimos "padronizar" o estado $\mathfrak{pe}_1(\mathfrak{C}, \beta)$ em estados, de passo único. Podemos tomar outro exemplo para enxergar isso novamente. No caso, será m-função $\mathfrak{e}$, *erase*, de um argumento, a qual recordamos logo a seguir.

Tabela 27 – Recapitulação da descrição a partir da *erase* de 1 argumento

Estado	**Símbolo**	**Operações**	**Estado final**
$\mathfrak{e}(\mathfrak{C})$	ə	R	$\mathfrak{e}_1(\mathfrak{C})$
	Não ə e não $\sqcup$	L	$\mathfrak{e}(\mathfrak{C})$
	Não $\sqcup$	L	$\mathfrak{e}(\mathfrak{C})$
$\mathfrak{e}_1(\mathfrak{C})$	Não $\sqcup$	R, E, R	$\mathfrak{e}_1(\mathfrak{C})$
	$\sqcup$		$\mathfrak{C}$

Fonte: Turing (1937)

O estado $\mathfrak{e}_1(\mathfrak{C})$ causa três operações. Poderíamos ter estados "de passo único" fazendo o trabalho dele, de pelo menos duas maneiras, indicadas abaixo.

Tabela 28 – Primeira versão de $\mathfrak{e}_1$ em passo único

Estado	**Símbolo**	**Operações**	**Estado final**
$\mathfrak{e}_{1,1}(\mathfrak{C})$	Não $\sqcup$	R	$\mathfrak{e}_{1,2}(\mathfrak{C})$
	$\sqcup$		$\mathfrak{C}$
$\mathfrak{e}_{1,2}(\mathfrak{C})$	Qualquer	E, R	$\mathfrak{e}_{1,1}(\mathfrak{C})$

Fonte: Autor (2024)

Tabela 29 – Segunda versão de $\mathfrak{e}_1$ em passo único

Estado	**Símbolo**	**Operações**	**Estado final**
$\mathfrak{e}_{1,1}(\mathfrak{C})$	Não $\sqcup$	R	$\mathfrak{e}_{1,2}(\mathfrak{C})$
	$\sqcup$		$\mathfrak{C}$
$\mathfrak{e}_{1,2}(\mathfrak{C})$	Qualquer	E	$\mathfrak{e}_{1,3}(\mathfrak{C})$
$\mathfrak{e}_{1,3}(\mathfrak{C})$	Qualquer	R	$\mathfrak{e}_{1,1}(\mathfrak{C})$

Fonte: Autor (2024)

Com esses exemplos, ficamos tentados a afirmar que qualquer um desses estados multipasso é sempre redutível a um conjunto finito de estados passo único. Se isso for verdade, então podemos utilizar informalmente os estados multipasso e ainda assim, no fundo, permaneceremos no universo das máquinas de Turing.

Para formalizarmos isto de que estamos falando, definiremos, a seguir, os termos utilizados.

Definição 3. Chamamos de **convenções de Turing** o conjunto das seguintes exigências para as operações imediatas a partir de um estado:

- lidar apenas com as operações de impressão de símbolo (inclusive apagar), e de deslocamento de uma unidade, para a esquerda ou para a direita;
- haver no mínimo nenhuma operação e no máximo duas;
- quando houver duas operações, a primeira é de impressão e a segunda é de deslocamento.

Definição 4. Chamamos de **estado passo único** a todo estado sob o qual são obedecidas as convenções de Turing.

E chamamos de **estado multipasso** a todo estado que provoca finitas operações, sob o qual não são obedecidas as convenções de Turing.

Dado isso, observemos duas coisas. Não afirmamos que sob os estados passo único a máquina também deve ter finitas operações pois, pelas convenções de Turing, isso fica evidente. E também que já estamos tratando a operação de apagar símbolo como a operação de imprimir $\sqcup$.

Agora, apresentamos convenções que, a princípio, são ainda mais restritivas que as de Turing. Enquanto Turing admite que um estado forneça menos de duas operações, o autor Sipser (2005) considera, para as máquinas de Turing, convenções distintas.

Definição 5. Chamamos de **convenções de Sipser** o conjunto das seguintes exigências para as operações imediatas a partir de um estado:

- lidar apenas com as operações de impressão de símbolo (inclusive apagar), e de deslocamento de uma unidade, para a esquerda ou para a direita;
- haver sempre duas operações;
- a primeira operação é de impressão e a segunda é de deslocamento.

Ou seja, nessas condições, a cabeça da máquina deve sempre imprimir algo e se deslocar para algum lado. Pode parecer que, seguindo isso, não obteremos alguns dos resultados, durante o processo da máquina, que as convenções de Turing permitiam.

Mas, como veremos em seguida, com as convenções de Sipser conseguimos obter tudo o que faríamos com as convenções de Turing. Para demonstrá-lo, recorreremos a uma notação que Turing utiliza quando ele demonstra a enumerabilidade

dos números computáveis. Quando a cabeça não se movimenta em um estado, Turing denota que esta é a operação “N”, que significa “*no move*” (sem movimento em inglês).

Sendo assim, para Turing, a operação do movimento da cabeça, que será ou R, ou L, ou N, sempre estará presente a partir de um estado passo único.

Lema 3.4.1. *Seja* $\mathfrak{q}'$ *um estado passo único de uma máquina de Turing, sob o qual ela sempre imprime algum símbolo, independentemente do símbolo escaneado. Se* $\mathfrak{q}'$ *prevê alguma operação N, então a ação a partir deste estado é substituível pela ação a partir de algum conjunto finito de estados segundo as convenções de Sipser.*

Demonstração. Tomemos uma máquina de Turing qualquer, de modo que possua alfabeto da fita usual e conjunto de estados possíveis também usual, tal que $\mathfrak{q}'$ é um desses estados e provoca transição como no enunciado.

Tomemos $i \in \mathbb{N}$, $0 \leq i \leq m-1$. Definimos que $\mathsf{D}_i \in \{\mathsf{R}, \mathsf{L}, \mathsf{N}\}$, que $l_i \in \{0, 1, \ldots, m-1\}$ e que $k_i \in \{0, 1, \ldots, n-1\}$. Com isso, esquematizamos, abaixo, as ações a partir do estado $\mathfrak{q}'$.

Tabela 30 – Descrição da máquina a partir do estado $\mathfrak{q}'$

Estado	Símbolo	Operações	Estado final
$\mathfrak{q}'$	α_0	$\mathsf{P}\alpha_{l_0}$, D_0	$\mathfrak{q}_{k_0}$
	α_1	$\mathsf{P}\alpha_{l_1}$, D_1	$\mathfrak{q}_{k_1}$
	$\vdots$	$\vdots$	$\vdots$
	α_{m-1}	$\mathsf{P}\alpha_{l_{m-1}}$, D_{m-1}	$\mathfrak{q}_{k_{m-1}}$

Fonte: Autor (2024)

Deste modo, para algum $i^* \in \{0, 1, \ldots, m-1\}$, temos que $\mathsf{D}_{i^*} = \mathsf{N}$.

Para nós demonstrarmos o lema em questão, definiremos um conjunto com novos estados para uma máquina de Turing. Tomaremos, para tal, todos os estados do conjunto $Q - \{\mathfrak{q}'\}$, junto dos estados $\mathfrak{q}^*$ e $\mathfrak{q}_0^*$, $\mathfrak{q}_1^*$, ..., $\mathfrak{q}_{n-1}^*$, sendo este últimos os novos estados. Assim sendo, definimos a ação da máquina sob o novo estado $\mathfrak{q}^*$ da maneira como segue logo abaixo.

Tabela 31 – Descrição da máquina a partir do estado $\mathfrak{q}^*$

Estado	Símbolo	Operações	Estado final
$\mathfrak{q}^*$	α_0	$\mathsf{P}\alpha_{l_0}$, $f(0)$	$g(0)$
	α_1	$\mathsf{P}\alpha_{l_1}$, $f(1)$	$g(1)$
	$\vdots$	$\vdots$	$\vdots$
	α_{m-1}	$\mathsf{P}\alpha_{l_{m-1}}$, $f(m-1)$	$g(m-1)$

Fonte: Autor (2024)

Apresentadas na tabela acima, f e g se tratam de funções. A primeira delas é

tal que

$$f(i) = \begin{cases} \mathsf{D}_i, & \text{se } \mathsf{D}_i \neq \mathsf{N}, \\ \mathsf{R}, & \text{se } \mathsf{D}_i = \mathsf{N} \end{cases},$$

e a segunda delas é tal que

$$g(i) = \begin{cases} \mathfrak{q}_{k_i}, & \text{se } \mathsf{D}_i \neq \mathsf{N} \text{ e } \mathfrak{q}_{k_i} \neq \mathfrak{q}', \\ \mathfrak{q}^*, & \text{se } \mathsf{D}_i \neq \mathsf{N} \text{ e } \mathfrak{q}_{k_i} = \mathfrak{q}', \\ \mathfrak{q}_i^*, & \text{se } \mathsf{D}_i = \mathsf{N} \end{cases}.$$

E também definimos as ações de nossa máquina de Turing sob os novos estados $\mathfrak{q}_i^*$ da seguinte maneira.

Tabela 32 – Descrição da máquina a partir dos estados $\mathfrak{q}_i^*$

Estado	**Símbolo**	**Operações**	**Estado final**
$\mathfrak{q}_i^*$	α_0	Pα_0, L	$h(i)$
	α_1	Pα_1, L	$h(i)$
	$\vdots$	$\vdots$	$\vdots$
	α_{m-1}	Pα_{m-1}, L	$h(i)$

Fonte: Autor (2024)

Temos, aqui, h como a função tal que

$$h(i) = \begin{cases} \mathfrak{q}_{k_i}, & \text{se } \mathfrak{q}_{k_i} \neq \mathfrak{q}', \\ \mathfrak{q}^*, & \text{se } \mathfrak{q}_{k_i} = \mathfrak{q}' \end{cases}.$$

Sob todos estes estados acima esquematizados, a máquina obedece às convenções de Sipser. Eles estão em quantidade finita. Eles têm o mesmo efeito de $\mathfrak{q}'$, de imprimir o que a máquina com ele imprimiria nos respectivos quadrados, e de não imprimir nada de novo nos demais. O movimento da cabeça acabará, após percorrer os respectivos estados, no mesmo quadrado em que acabaria originalmente.

E a mudança de estados, ao final, também é a mesma, exceto no caso em que mudava para $\mathfrak{q}'$. Agora terminará mudando para $\mathfrak{q}^*$. Mas justamente, queremos substituir $\mathfrak{q}'$ pelo conjunto de novos estados. E eles a partir deles são cumpridas todas as outras ações que $\mathfrak{q}'$ se propunha a provocar.

Então, em vez de seguirmos em direção a $\mathfrak{q}'$, devemos, agora, seguir ao novo estado com que é "emulado" o início das ações de $\mathfrak{q}'$. Este estado, aqui, é $\mathfrak{q}^*$.

Com tudo isso, observemos ainda o seguinte. Podemos utilizar o mesmo número de estados restantes do conjunto Q junto ao conjunto de novos estados. Apenas tendo o cuidado de trocar uma coisa nas descrições de suas ações subsequentes. Onde o seu estado sucessor era originalmente $\mathfrak{q}'$, agora será $\mathfrak{q}^*$. ■

Lema 3.4.2. *Admitindo a presença da operação N, todos os estados passo único podem ter suas ações substituídas pelas ações de um conjunto finito de estados que obedeçam às convenções de Sipser.*

Demonstração. Provemos, primeiramente, um resultado anterior. Seja $\mathfrak{q}''$ um estado passo único com o qual, para algum símbolo escaneável, a máquina faça menos que duas operações. Então demonstraremos e usaremos o fato de que a ação deste estado pode ser substituída pela ação de algum conjunto finito de estados que obedeça às convenções de Sipser.

Seja uma máquina de Turing, com alfabeto da fita usual e conjunto de estados possíveis também usual, tal que $\mathfrak{q}''$ é um desses estados. Então, para algum dos símbolos que a máquina escanear a partir desse estado, ela fará apenas uma ou nenhuma operação. E, se fizer nenhuma operação, podemos considerar, no lugar, que ele está fazendo a operação N. Assim, este estado sempre fará ao menos uma operação.

Chamemos de $\mathfrak{q}^*$ o estado pelo qual iremos, a princípio, substituir $\mathfrak{q}''$.

Suponhamos que a operação da máquina a partir de $\mathfrak{q}''$, e sobre este símbolo escaneável, era a impressão de algum símbolo. Então basta que a operação a partir do mesmo símbolo escaneável, com a máquina estando sobre o estado $\mathfrak{q}^*$, seja a mesma impressão, seguida pela operação N.

Suponhamos, agora, que a operação da máquina a partir de $\mathfrak{q}''$, sobre este símbolo escaneável, era R, ou L, ou N. Então basta que a operação a partir do mesmo símbolo escaneável, com a máquina estando em $\mathfrak{q}^*$, seja a impressão do próprio símbolo escaneado, seguida da operação original de deslocamento, fosse ela R, ou fosse L, ou fosse N.

O estado sucessor de $\mathfrak{q}^*$, a partir do presente símbolo escaneável, se era o próprio estado $\mathfrak{q}''$, agora será o estado $\mathfrak{q}^*$. E, se era um estado de Q que não $\mathfrak{q}''$, continua sendo o estado sucessor original.

Onde havia já duas operações sob o estado $\mathfrak{q}''$, sob o estado $\mathfrak{q}^*$ continuarão havendo. Assim, podemos ver que as operações substitutas são equivalentes às originais. E a mudança de estados também. Inclusive a mudança de volta para $\mathfrak{q}^*$, dado que é o estado que está substituindo devidamente o estado $\mathfrak{q}''$.

Agora, basta fazermos as adaptações nas transições dos estados restantes de Q. Trata-se apenas de trocar a descrição de suas ações subsequentes onde tinha como estado sucessor o $\mathfrak{q}''$, para colocar $\mathfrak{q}^*$ em seu lugar.

No entanto, o novo estado não obedece às convenções de Sipser. Apenas pelo fato de que está sendo levada em conta a operação N. Mas, pelo lema 3.4.1, podemos substituir o estado em questão por um conjunto finito de novos estados, os quais obedecem às convenções referidas.

Com isso, o fato está provado. E utilizando ele, finalizaremos a nossa demons-

tração.

Olhemos para o estado $\mathfrak{q}'$. No caso onde, a partir dele, não há operação ou há uma só, provamos que a substituição dele por finitos estados com as convenções de Sipser é possível. E, nestes estados substitutos, após a cabeça ler o símbolo do quadrado escaneado original, onde eram descritas já antigamente uma dupla de operações, assim continua sendo.

E, por fim, em um estado passo único onde só houvesse duplas de operações, já se está obedecendo às convenções de Sipser. ■

Enfim, tendo isso em mente, podemos nos dirigir finalmente ao resultado central da seção.

Teorema 3.4.3. *Aquilo que um estado multipasso qualquer provoca na máquina de Turing sempre pode ser substituído pela ação tida a partir de algum conjunto finito de estados passo único.*

Demonstração. Seja $\mathfrak{q}'$ um estado multipasso qualquer. A máquina de Turing possui alfabeto da fita usual e conjunto de estados possíveis também usual. O estado $\mathfrak{q}'$, com isso, pertence a esse último conjunto.

Sendo assim, as ações dessa máquina de Turing a partir do estado $\mathfrak{q}'$ funciona como descrito a seguir.

Tabela 33 – Descrição da máquina a partir do estado multipasso $\mathfrak{q}'$

Est.	Símb.	Operações	Est. final
$\mathfrak{q}'$	α_0	$\mathsf{op}_{0,1}, \mathsf{op}_{0,2}, \ldots, \mathsf{op}_{0,n_0}$	$\mathfrak{q}_{k_0}$
	α_1	$\mathsf{op}_{1,1}, \mathsf{op}_{1,2}, \ldots, \mathsf{op}_{1,n_1}$	$\mathfrak{q}_{k_1}$
	$\vdots$	$\vdots$	$\vdots$
	α_{m-1}	$\mathsf{op}_{m-1,1}, \mathsf{op}_{m-1,2}, \ldots, \mathsf{op}_{m-1,n_{m-1}}$	$\mathfrak{q}_{k_{m-1}}$

Fonte: Autor (2024)

Para $i, j \in \mathbb{N}$, $0 \leq i \leq m-1$, temos que n_i é um natural positivo qualquer, e $\mathsf{op}_{i,j}$ significa a j-ésima operação da máquina no estado $\mathfrak{q}'$ após a cabeça escanear o símbolo α_i. Ainda, o número k_i é um natural tal que $0 \leq k_i \leq n-1$.

Cada operação destas acima pode ser qualquer uma, ou seja, imprimir qualquer símbolo, ou deslocar uma unidade para a direita, ou uma para a esquerda. Ou ainda, uma $\mathsf{op}_{i,j}$ pode ser operação nenhuma, mas somente se, para aquele símbolo escaneável α_i, ela for a única operação presente. Ou seja, sob este símbolo, o estado causa operação nenhuma na máquina.

Sendo assim, a partir de cada um dos símbolos do alfabeto da fita Γ, definamos um conjunto de novos estados para nossa máquina de Turing. Definimos a ação de nossa máquina a partir do estado $\mathfrak{q}'_0$ conforme descrevemos abaixo, na tabela 34.

Tabela 34 – Descrição da máquina a partir do estado $\mathfrak{q}'_0$

Estado	Símbolo	Operações	Estado final
$\mathfrak{q}'_0$	α_0	$\mathrm{op}_{0,1}$	$\mathfrak{q}'_{0,2}$
	α_1	$\mathrm{op}_{1,1}$	$\mathfrak{q}'_{1,2}$
	$\vdots$	$\vdots$	$\vdots$
	α_{m-1}	$\mathrm{op}_{m-1,1}$	$\mathfrak{q}'_{m-1,2}$

Fonte: Autor (2024)

Definimos, para $l \in \mathbb{N}$, $2 \leq l < n_i$, a ação da máquina a partir dos estados $\mathfrak{q}'_{i,l}$ da seguinte maneira.

Tabela 35 – Descrição da máquina a partir do estado $\mathfrak{q}'_{i,l}$

Estado	Símbolo	Operações	Estado final
$\mathfrak{q}'_{i,l}$	Qualquer	$\mathrm{op}_{i,l}$	$\mathfrak{q}'_{i,l+1}$

Fonte: Autor (2024)

E definimos a ação a partir dos estados $\mathfrak{q}'_{i,n_i}$ da seguinte maneira.

Tabela 36 – Descrição da máquina a partir do estado $\mathfrak{q}'_{i,n_i}$

Estado	Símbolo	Operações	Estado final
$\mathfrak{q}'_{i,n_i}$	Qualquer	op_{i,n_i}	$f(i)$

Fonte: Autor (2024)

Temos, aqui, f como a função tal que

$$f(i) = \begin{cases} \mathfrak{q}_{k_i}, & \text{se } \mathfrak{q}_{k_i} \neq \mathfrak{q}', \\ \mathfrak{q}'_0, & \text{se } \mathfrak{q}_{k_i} = \mathfrak{q}' \end{cases}.$$

Tomando estes estados, com o início de sua ação em $\mathfrak{q}'_0$, eles reproduzem, na máquina, exatamente as mesmas operações que ela faria a partir de $\mathfrak{q}'$, na mesma ordem em que ocorreriam, e nenhuma outra a mais.

Provoca, ao final, a mudança para os mesmos estados, exceto no caso em que mudaria para $\mathfrak{q}'$. Aí, a mudança é para $\mathfrak{q}'_0$. Só que esta é a mudança devida agora, visto que queremos substituir o estado $\mathfrak{q}'$ por este novo conjunto de estados, com início da ação em $\mathfrak{q}'_0$. Assim, os novos estados, juntos, possuem o mesmo efeito que o antigo.

Conforme descritos, tais estados se encontram em quantidade finita. E todos são estados passo único. Afinal todos possuem, em suas operações, exatamente um elemento da forma $\mathrm{op}_{i,j}$, o qual é, ou nenhuma operação, ou é uma operação de impressão, ou uma de movimento.

Por fim, fazemos as adaptações nos estados restantes de Q. Isto é, simplesmente, nas descrições de suas ações subsequentes, onde o estado sucessor era o $\mathfrak{q}'$, colocamos $\mathfrak{q}'_0$ no lugar. ■

Corolário 3.4.4. *Aquilo que um estado multipasso qualquer provoca na máquina de Turing sempre pode ser substituído pela ações provocadas por algum conjunto finito de estados segundo as convenções de Sipser.* ■

3.5 UMA DEFINIÇÃO ALGÉBRICA

Após conhecermos um tanto as máquinas de Turing, nesta seção traremos uma nova formalização de sua definição. Como pudemos enxergar pelas distintas convenções, de Turing e de Sipser, há algo a inicialmente notarmos. Que é possível, a partir de variações razoáveis da definição original para a máquina de Turing, gerar uma classe equivalente de máquinas.

Seria interessante agora, entendermos o conceito de linguagem, no contexto em que estamos inseridos. Para isso comecemos por conceitos preliminares, tal como apresenta Sipser na bibliografia aqui referenciada.

Chamamos de **alfabeto** todo conjunto finito e não-vazio. E dizemos que os seus elementos são os seus **símbolos**. Tal como nos alfabetos da fita com que lidamos anteriormente, costumamos representar estes conjuntos por uma letra grega maiúscula.

Chamamos de **cadeia sobre um alfabeto** a toda sequência finita de seus símbolos. É comum que se escreva as cadeias por meio da sucessão dos respectivos símbolos, sem separação deles por vírgula. Sendo w uma cadeia de um dado alfabeto, dizemos que o **comprimento de** w é o número de símbolos sequenciados, e o denotamos por $|w|$.

Por exemplo, do alfabeto $\Sigma_1 = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f}\}$, podemos tirar a cadeia fada. E daí, $|\mathsf{fada}| = 4$.

Chamamos de **cadeia vazia** a cadeia cujo comprimento é igual a zero, e a denotamos por ε. Quando uma cadeia w tem comprimento l, podemos denotá-la por $w = w_1 w_2 \dots w_l$, onde, para $i = 1, 2, \dots, l$, w_i é o i-ésimo elemento da sequência que a compõe.

E, por fim, chamamos de **linguagem** a todo conjunto de cadeias de um mesmo alfabeto.

Tomando o mesmo alfabeto Σ_1, montamos as linguagens $A = \{\mathsf{dbf}, \mathsf{ac}, \mathsf{eedadcb}\}$, $B = \{\mathsf{a}, \varepsilon, \mathsf{dabccc}, \mathsf{ee}, \mathsf{ac}\}$, $C = \{\varepsilon, \mathsf{d}, \mathsf{de}, \mathsf{ded}, \mathsf{dede}, \mathsf{deded}, \dots\}$ e outras inúmeras.

Apresentado isto, tomemos ainda dois outros conceitos, um usado para cadeias e outro para linguagens. O primeiro define que, dadas duas cadeias $x = x_1x_2 \dots x_{l_1}$ e $y = y_1y_2 \dots y_{l_2}$, então a **concatenação** de x e y é a cadeia $xy = x_1x_2 \dots x_{l_1}y_1y_2 \dots y_{l_2}$.

O segundo define que, dada uma linguagem A, temos a operação **estrela**, segundo a qual obtemos a linguagem "A estrela", que se trata da nova linguagem $A^* = \{\varepsilon\} \cup \{w_1w_2 \dots w_k; k \geq 1 \text{ e } w_1, w_2, \dots w_k \in A\}$.

Assim, com o alfabeto $\Sigma_2 = \{0, 1\}$, e a linguagem $D = \{00, 11\}$, temos que $D^* = \{\varepsilon, 00, 11, 0000, 0011, 1100, 1111, 000000, 000011, 001100, \dots\}$.

Observemos que todo alfabeto é também uma linguagem, a qual toma as cadeias unitárias dos símbolos do próprio alfabeto. Então faz sentido falarmos, por exemplo, de Σ_2^*, igual a $\{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 100, 011, \dots\}$.

Com tudo isso, trazemos agora uma definição algébrica para as máquinas de Turing. Com ela, ficam delimitadas as cadeias a partir das quais uma máquina de Turing iniciará sua atividade.

Definição 6. Sendo Q, Σ e Γ conjuntos finitos, chamamos de **máquina de Turing** toda 7-upla $(Q, \Sigma, \Gamma, \delta, \mathfrak{q}_0, \mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}})$ tal que

1. Q é o **conjunto de estados**;
2. Σ é o **alfabeto de entrada** da máquina, que não possui o símbolo ␣;
3. Γ é o **alfabeto da fita**, tal que $\text{␣} \in \Gamma$ e $\Sigma \subset \Gamma$;
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\text{R}, \text{L}\}$ é a **função de transição**;
5. $\mathfrak{q}_0 \in Q$ é o **estado inicial** da máquina;
6. $\mathfrak{q}_{\text{aceita}} \in Q$ é o chamado **estado de aceitação** da máquina;
7. e $\mathfrak{q}_{\text{rejeita}} \in Q$ é o chamado **estado de rejeição** da máquina, de modo que $\mathfrak{q}_{\text{rejeita}} \neq \mathfrak{q}_{\text{aceita}}$.

Trata-se de uma definição mais genérica. Ela engloba as máquinas de Turing tal como apresentamos no início desta dissertação, mas também muitas outras classes de máquinas. Ela não se limita ao dever de imprimir sequências de algarismos. Seus alfabetos, em verdade, nem precisam ter algarismos. Agora explicaremos como ela deve ser encarada.

Já estamos familiarizados com os conjuntos Q, Γ e com o estado $\mathfrak{q}_0$, o qual se trata do estado com que a máquina começa sua atividade. Como podemos ver, a máquina, aqui, não precisa se restringir a zero e um como seus F-símbolos. Em verdade, não é necessário sequer falar de números em si. Mas vamos entender tal definição em torno do contexto em que estamos tratando as máquinas desde o começo.

Até agora lidávamos com máquinas de Turing cuja fita começava em branco. Mas nem sempre é assim. Veremos, quando falarmos das máquinas de Turing universais, exemplos disso. Mas por ora tomemos nota do seguinte: uma máquina de Turing pode começar com uma cadeia de dado comprimento impressa em sua fita.

Essa cadeia não tem símbolos em branco. Quando a percorremos com o olhar e chegamos a um quadrado em branco, eis aí o sinal de que acabamos de ultrapassar seu fim. É a partir disso que tomamos o alfabeto de entrada Σ. Tal cadeia pertence justamente ao conjunto Σ^*.

Segundo a definição, o alfabeto da fita contém o alfabeto de entrada. Mas o que os diferencia não precisa ser apenas o símbolo $\sqcup$. Podem haver ainda outros símbolos exclusivos do alfabeto da fita. Como também podem não haver, e o espaço vazio ser a única coisa que os distinga.

Na antiga definição, com máquinas de Turing que sempre começavam com a entrada vazia, acabava que cada máquina de Turing era limitada a um único trabalho, fosse de computar uma determinada sequência ou outro. Agora, com as cadeias do alfabeto de entrada, a máquina consegue mostrar uma variedade de caminhos percorridos ao longo de sua computação, como também de resultados a fornecer.

Convencionamos que a fita tem um começo de fato, à esquerda, e não tem fim para a direita. É no seu primeiro quadrado que será impresso o primeiro símbolo da cadeia de entrada. Caso a cabeça da máquina fosse para a esquerda após escanear o primeiro quadrado, ela segue posicionada sobre o primeiro quadrado, para então escanear seu símbolo no estado em que a máquina agora se encontrar.

A função de transição δ é responsável pelo que a máquina fará entre os estados e os símbolos escaneados. Pois, quando a cabeça se encontra no estado $\mathfrak{q}$ e escaneia o símbolo α, seguindo as convenções de Sipser, ela imprimirá um símbolo α', se movimentará uma unidade para a esquerda ou direita, e passará para um estado $\mathfrak{q}'$.

E essa transição é a descrição do significado da aplicação δ, na qual acontece que, ou $\delta(\mathfrak{q}, \alpha) = (\mathfrak{q}', \alpha', \mathsf{L})$, ou $\delta(\mathfrak{q}, \alpha) = (\mathfrak{q}', \alpha', \mathsf{R})$.

Por fim, temos os estados de aceitação e de rejeição. Quando a máquina entra em qualquer um desses estados, ela para a sua atividade. Basicamente eles vão dizer, respectivamente, se a máquina aceita ou se ela desconsidera determinadas cadeias. Em que contexto? Falamos das cadeias que estiverem impressas na fita no início de sua atividade.

Se dada cadeia inicial $w \in \Sigma^*$, com a atividade da máquina, leva a sucessão de estados até o estado $\mathfrak{q}_{\text{aceita}}$ em algum momento, dizemos que a máquina **aceita** a entrada w.

Poderíamos dizer que determinadas cadeias, a partir do nosso alfabeto de entrada, "satisfazem" aquilo para que uma particular máquina de Turing foi construída. Estamos falando de todas as cadeias w que nos levam até o estado $\mathfrak{q}_{\text{aceita}}$.

E outras cadeias do mesmo alfabeto não só não satisfazem, mas são consideradas “erradas” dentro daquela máquina de Turing. Tratam-se das cadeias de entrada que nos levam até o estado $\mathfrak{q}_{\text{rejeita}}$. E quando uma cadeia inicial $w \in \Sigma^*$ leva a sucessão de estados da máquina até o estado $\mathfrak{q}_{\text{rejeita}}$, dizemos que a máquina **rejeita** a entrada w.

Exemplo 6. Sejam os conjuntos $Q = \{\mathfrak{q}_0, \mathfrak{q}_1, \mathfrak{q}_2, \mathfrak{q}_3, \mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}}\}$, $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ e $\Gamma = \{\textvisiblespace, \mathsf{a}, \mathsf{b}, \mathsf{c}, \#\}$. Seja também a função $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathsf{R}, \mathsf{L}\}$, que é definida por

$$\delta(\mathfrak{q}, \alpha) = \begin{cases} (\mathfrak{q}_1, \mathsf{b}, \mathsf{R}), & \text{se } \mathfrak{q} = \mathfrak{q}_0 \text{ e } \alpha \neq \mathsf{c}, \\ (\mathfrak{q}_2, \mathsf{b}, \mathsf{L}), & \text{se } \mathfrak{q} = \mathfrak{q}_0 \text{ e } \alpha = \mathsf{c}, \\ (\mathfrak{q}_1, \mathsf{a}, \mathsf{R}), & \text{se } \mathfrak{q} = \mathfrak{q}_1 \text{ e } \alpha \neq \mathsf{a}, \\ (\mathfrak{q}_3, \#, \mathsf{R}), & \text{se } \mathfrak{q} = \mathfrak{q}_1 \text{ e } \alpha = \mathsf{a}, \\ (\mathfrak{q}_{\text{aceita}}, \mathsf{c}, \mathsf{R}), & \text{se } \mathfrak{q} = \mathfrak{q}_2 \text{ e } \alpha \neq \mathsf{a}, \\ (\mathfrak{q}_{\text{rejeita}}, \textvisiblespace, \mathsf{L}), & \text{se } \mathfrak{q} = \mathfrak{q}_2 \text{ e } \alpha = \mathsf{a}, \\ (\mathfrak{q}_1, \mathsf{c}, \mathsf{R}), & \text{se } \mathfrak{q} = \mathfrak{q}_3 \text{ e } \alpha \neq \mathsf{b}, \\ (\mathfrak{q}_{\text{rejeita}}, \mathsf{b}, \mathsf{R}), & \text{se } \mathfrak{q} = \mathfrak{q}_3 \text{ e } \alpha = \mathsf{b}, \\ (\mathfrak{q}_2, \mathsf{c}, \mathsf{R}), & \text{se } \mathfrak{q} = \mathfrak{q}_{\text{aceita}}, \\ (\mathfrak{q}_3, \mathsf{a}, \mathsf{R}), & \text{se } \mathfrak{q} = \mathfrak{q}_{\text{rejeita}} \end{cases}.$$

Montamos, daí, a máquina de Turing $\mathcal{M}^{\spadesuit} = (Q, \Sigma, \Gamma, \delta, \mathfrak{q}_0, \mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}})$. A partir da função de transição que possui, podemos dar a esta máquina a descrição tabular indicada mais abaixo, na tabela 37.

Certamente há um conjunto de cadeias de entrada que levam esta máquina ao estado de aceitação. Como também um conjunto de entradas que a levam ao estado de rejeição. Podemos dizer que eles estão bem-definidos, pois mesmo que não houvesse quaisquer cadeias em algum deles, ainda assim seria o caso de ser igual ao conjunto vazio.

Mas será o caso de não haver estes ou aqueles elementos para esta máquina? E serão estas duas possibilidades as únicas existentes? Confiramos isto, a partir de determinadas entradas.

Tomemos o exemplo em que a fita da presente máquina começa com a cadeia de entrada caba. A cabeça começa seu trabalho, então, escaneando o símbolo c e no estado $\mathfrak{q}_0$. Pela descrição da máquina, a cabeça vai imprimir b no lugar de c e mover-se uma unidade para a esquerda. A máquina muda para o estado $\mathfrak{q}_2$.

Só que estamos no início da fita. Não podemos andar mais para a esquerda, então permanecemos no primeiro quadrado, agora com b impresso. A cabeça, agora, o escaneia, com o estado $\mathfrak{q}_2$. Ela então imprime c no quadrado escaneado, move-se

Tabela 37 – Descrição tabular da máquina $\mathcal{M}\spadesuit$

Estado	Símbolo	Operações	Estado final
q_0	Não c	Pb, R	q_1
	c	Pb, L	q_2
q_1	Não a	Pa, R	q_1
	a	P#, R	q_3
q_2	Não a	Pc, R	q_{aceita}
	a	P␣, L	$q_{rejeita}$
q_3	Não b	Pc, R	q_1
	b	Pb, R	$q_{rejeita}$
q_{aceita}	Qualquer	Pc, R	q_2
$q_{rejeita}$	Qualquer	Pa, R	q_3

Fonte: Autor (2024)

uma unidade para a direita e a máquina muda para o estado q_{aceita}.

Com o que acabamos de ver, a máquina finaliza a atividade deixando a cadeia caba impressa na fita. E podemos afirmar que tal máquina aceita a entrada caba.

Sabemos a imagem da função de transição a partir do estado de aceitação com os símbolos escaneáveis. Mas em verdade ela nunca interfere no trabalho da máquina, já que ele é finalizado quando ela chega neste estado. O mesmo vale para o estado de rejeição.

Vejamos agora o trabalho da máquina a partir da cadeia de entrada babca. A cabeça escaneia o símbolo b em q_0. Daí imprime b no mesmo lugar e move-se uma unidade para a direita. A máquina muda para o estado q_1.

Agora a cabeça escaneia o símbolo a. Com isso, imprime # no lugar e se move uma unidade para a direita. A máquina passa para o estado q_3. A cabeça escaneia outro símbolo b, e daí imprime b no mesmo lugar e se move uma unidade para a direita. A máquina, então, passa para o estado $q_{rejeita}$, e para a sua atividade.

Deixou impressa na fita a cadeia b#bca. E, ao final, vemos que tal máquina rejeita a entrada babca. Podemos tomar tantos outros exemplos de entradas as quais nossa máquina não aceita. Como também de entradas que ela aceita. E ainda sobra uma pergunta. E se a máquina nunca chegar aos estados q_{aceita} e $q_{rejeita}$?

Realmente, isso pode acontecer. Para isso, olhemos para o exemplo da entrada abb. A máquina, no estado q_0, tem sua cabeça escaneando a. Imprime b no lugar e move-se uma unidade para a direita. A máquina agora se encontra no estado q_1.

Ela escaneia o primeiro símbolo b original, a cabeça imprime a no lugar e move-se uma unidade para a direita. A máquina ainda encontra-se no estado q_1. Escaneia o segundo b original, a cabeça imprime a e move-se para a direita de novo.

Volta para q_1, do qual não sairá mais. Afinal, a partir de agora, escaneará infinitos símbolos ␣. A partir deles, sempre imprimirá a no lugar, anda para a direita e retorna a q_1. Nunca vai mudar para q_{aceita} e $q_{rejeita}$. Continuará sua atividade sem parar.

Imprimirá na fita a cadeia infinita baaaaaaa . . .

Agora, como pudemos ver, a nossa máquina aceitava uma cadeia de entrada. É de se esperar que esta máquina aceite outras cadeias de Σ^*, e exemplos de cadeias que ela aceita são c, ca, cb, cc, caa, cba, cca, cab, entre várias outras. E, neste caso, podemos facilmente identificar o padrão de tais cadeias de entrada.

Para que se chegue ao estado $\mathfrak{q}_{\text{aceita}}$, é necessário passar pelo estado $\mathfrak{q}_2$ escaneando um símbolo $\alpha \neq$ a logo antes. Ora, o único estado que antecede $\mathfrak{q}_2$ de fato é $\mathfrak{q}_0$. E para que a sucessão seja esta correspondente, em $\mathfrak{q}_0$, a cabeça da máquina deve escanear o símbolo c.

Já que nenhum outro estado faz com que a máquina retorne a $\mathfrak{q}_0$, significa que tudo isto só pode acontecer no início da atividade da máquina. E, logo após ela escanear o símbolo c do início, a cabeça tentará andar para a esquerda, mas se encontrará com o limite da fita. Dado isso, o próximo passo é sempre com a cabeça no primeiro quadrado, tendo um b ali impresso. Portanto, a máquina irá, após entrar no estado $\mathfrak{q}_2$, para o estado de aceitação.

Com isso, é visível que as entradas que são aceitas pela nossa máquina são exatamente todas aquelas que começam com o símbolo c. □

Definição 7. Dizemos que uma máquina de Turing $\mathcal{M}$ **reconhece** a linguagem A se, e somente se A é o conjunto das cadeias de entrada que $\mathcal{M}$ aceita. Também dizemos que A é **a linguagem de** $\mathcal{M}$, o que denotamos por $L(\mathcal{M}) = A$.

E afirmamos que uma dada linguagem é **Turing-reconhecível** se, e somente se existe alguma máquina de Turing que a reconhece.

Uma máquina sempre possui uma linguagem. Pois mesmo que ela não aceite entrada alguma, nesta situação teremos que $L(\mathcal{M}) = \varnothing$. E, no caso exemplificado acima, temos que $L\left(\mathcal{M}^{\spadesuit}\right) = \{w_1 w_2 \dots w_k \in \Sigma^*;\ k \geq 1 \text{ e } w_1 = \text{c}\}$.

Quando uma máquina de Turing não chega nunca ao estado de aceitação e nem ao de rejeição, dizemos que ela **entrou em loop**. Nessa condição, ela poderia sim estar imprimindo uma cadeia infinita. Mas poderia também, por exemplo, ter ficado "presa" em um conjunto finito de mesmos quadrados da fita, dos quais não consegue sair devido à sua descrição e aos símbolos que está escaneando. Como pode, também, acabar envolvida fazendo coisas mais complicadas.

Por vezes, é conveniente que tenhamos uma máquina de Turing que acabe por apenas chegar ou ao estado de aceitação ou ao de rejeição. Afinal, após uma longa passagem de tempo, pode ser difícil saber se uma dada máquina está em loop ou se ela vai chegar a um dos estados finais.

Definição 8. Uma máquina de Turing que nunca entra em loop é chamada de um **decisor**. Quando um decisor $\mathcal{M}$ reconhece uma linguagem A, dizemos que $\mathcal{M}$ **decide** a linguagem A.

E chamamos uma dada linguagem de **decidível** se, e somente se existe alguma máquina de Turing que a decide.

Ou seja, não basta que uma máquina de Turing aceite apenas e exatamente as cadeias da linguagem A para que se afirme que A é uma linguagem decidível. Sendo Σ o alfabeto da fita, é necessário também que haja alguma máquina de Turing que, além de reconhecer A, concomitantemente, rejeite todas as cadeias de $\Sigma^* - A$. Não há entrada sob a qual tal máquina entrará em loop.

No exemplo que demos acima, a linguagem $L\left(\mathcal{M}\spadesuit\right)$ é Turing-reconhecível. Talvez queiramos afirmar que ela não é decidível, mas, a princípio, não o sabemos. Pois sabemos que a máquina $\mathcal{M}\spadesuit$ não a decide. Mas isso não quer dizer que não exista alguma outra máquina de Turing que decide esta linguagem.

Sendo assim, entendamos agora qual é a classe de máquinas de Turing que se encaixam na descrição que demos no início desta dissertação. Recordemos, pois, a definição 1, acrescida da condição de colocarmos no início da fita os símbolos ə.

Do jeito que Turing as definiu, sem tratarmos ainda da máquina universal, o alfabeto de entrada é o conjunto vazio, pois a fita inicia sem símbolos visíveis. O alfabeto da fita, tido por $\Gamma^{\mathcal{T}}$, sempre possui ao menos os símbolos 0, 1, ə e claro, ␣.

O conjunto de estados, $Q^{\mathcal{T}} = \{\mathfrak{q}_0, \mathfrak{q}_1, \ldots, \mathfrak{q}_{n-1}\}$, com $n \geq 5$, por si só, não tem nada de especial. Sendo $\mathfrak{q}_0$ o estado inicial, e fixando $\mathfrak{q}_{n-2} = \mathfrak{q}_{\text{aceita}}$ e $\mathfrak{q}_{n-1} = \mathfrak{q}_{\text{rejeita}}$, a função de transição $\delta^{\mathcal{T}}$ pode ser descrita como abaixo.

Teremos $\delta^{\mathcal{T}}(\mathfrak{q}_0, ␣) = (\mathfrak{q}_1, \text{ə}, \mathsf{R})$ e $\delta^{\mathcal{T}}(\mathfrak{q}_1, ␣) = (\mathfrak{q}_2, \text{ə}, \mathsf{R})$. Para $\mathsf{X} \in \{\mathsf{R}, \mathsf{L}\}$ e $i, j \in \{2, 3, \ldots, n-3\}$, ocorre que $\delta^{\mathcal{T}}(\mathfrak{q}_i, \alpha) = (\mathfrak{q}_j, \alpha', \mathsf{X})$; $\delta^{\mathcal{T}}(\mathfrak{q}_2, ␣) = (\mathfrak{q}_j, \beta, \mathsf{X})$ onde $\beta \neq \text{ə}$; e $\delta^{\mathcal{T}}(\mathfrak{q}_i, \alpha) = (\mathfrak{q}_j, \alpha', \mathsf{X})$, onde $\alpha = \text{ə}$ se, e somente se $\alpha' = \text{ə}$ e $\mathsf{X} = \mathsf{R}$. Também permanece estabelecido que os dígitos impressos nos F-quadrados não serão apagados, e que neles serão impressos da esquerda para a direita, sem pular nenhum de tais quadrados. Cumpridas estas condições, há liberdade para se definir as respectivas imagens.

Fixando $\alpha_0 = ␣$, $\alpha_1 = 0$, $\alpha_2 = 1$ e $\alpha_3 = \text{ə}$, temos $\Gamma^{\mathcal{T}} = \{\alpha_0, \alpha_1, \ldots, \alpha_{m-1}\}$, com $m \geq 4$. A partir de α_4, podemos considerar os símbolos como marcadores. É então à classe das máquinas $\mathcal{M}^{\mathcal{T}} = \left(Q^{\mathcal{T}}, \varnothing, \Gamma^{\mathcal{T}}, \delta^{\mathcal{T}}, \mathfrak{q}_0, \mathfrak{q}_{n-2}, \mathfrak{q}_{n-1}\right)$ que nos referíamos mais cedo.

Observação 4. Dentro dessa classe de máquinas, claramente poderíamos adotar diferentes marcadores para um mesmo índice em diferentes alfabetos. Mas dois alfabetos da fita de mesma cardinalidade, para máquinas com mesmo $Q^{\mathcal{T}}$, e com $\delta^{\mathcal{T}}$ estruturada da mesma forma, acabam cumprindo sempre o mesmo papel.

Afinal não é em si o formato particular de um marcador que lhe garante que terá um determinado papel, mas simplesmente o fato de ele ser diferente dos demais. Se o marcador x for escolhido como o símbolo α_7, a máquina, a partir dele, fará um

trabalho equivalente ao que faria se α_7 fosse igual a $\mathbf{y}$. Então, essencialmente, nessa situação, o que importa é a cardinalidade de $\Gamma^{\mathcal{T}}$.

Enfim, podemos notar algumas coisas. As máquinas desta classe nunca chegam aos estados de aceitação e de rejeição. Podem ser circulares ou não-circulares. Nelas, o símbolo ə só será impresso em dois quadrados no começo, à esquerda do restante das impressões. Elas podem usar a quantidade finita de marcadores que quiser. O mesmo para a quantidade de estados.

E claro, sendo o alfabeto de entrada vazio, só haverá uma única cadeia a ser impressa na fita para cada uma dessas máquinas. Claro que, propriamente falando, um conjunto vazio, por definição, não é um alfabeto. Mas basta que compreendamos que estamos chamando de *alfabeto de entrada* aquele conjunto da 7-upla que cumpre os requisitos para este papel.

3.6 COMPUTÁVEIS E ENUMERÁVEIS

Agora faremos a demonstração de algo antes comentado. O fato de que os números computáveis são enumeráveis. Sendo o conjunto dos números reais não-enumerável, isso significa que nem todo número real é computável.

Já sabemos que os números computáveis são infinitos, pois já provamos que todos os racionais são computáveis. Isto significa que há uma bijeção do conjunto dos números computáveis com o conjunto dos números naturais.

Esta nossa afirmação será dada como corolário a partir das máquinas $\mathcal{M}^{\mathcal{T}}$. Antes, provaremos que, tomando uma quantidade enumerável de alfabetos da fita, as máquinas de Turing $\mathcal{M}$ existem em quantidade enumerável. Consequentemente, nessas condições, as máquinas da classe das $\mathcal{M}^{\mathcal{T}}$ são enumeráveis. Isto implicará na enumerabilidade das sequências computáveis.

E isto, por sua vez, implicará na enumerabilidade dos próprios números computáveis. Por quê? Basta retornarmos à definição deles. A partir dela, fazemos a seguinte observação. Um dado número x é computável, isto é, possui uma sequência computável equivalente à sua parte fracionária, se, e somente se, para todo inteiro k, ocorre que $k + x$ também é computável. Assim, se $3/5 = 0,6$ é computável, também o são o número $1,6$, o número $2,6$, o $5,6$, o $11,6$, o $234,6$, o $-0,4$, o $-1,4$, o $-3,4$, o $-20,4$, e tantos outros.

Vemos que descobrir uma sequência computável, significa descobrir uma quantidade infinita e enumerável de números computáveis. E daí, o conjunto de todos os

números computáveis se trata de

$$\bigcup_{x\in[0,1) \text{ é computável}} \{k+x;\, k\in\mathbb{Z}\}.$$

E isto concluirá a prova, visto novamente que a reunião enumerável de enumeráveis é um conjunto enumerável. Agora, prossigamos, com o resultado e sua demonstração.

Teorema 3.6.1. *Se tomamos uma quantidade enumerável de alfabetos da fita, então a quantidade de máquinas de Turing existentes a partir deles é enumerável.*

Demonstração. Basta que provemos que a quantidade de máquinas geradas por qualquer um desses alfabetos sempre é enumerável, pois daí o conjunto de todas tais máquinas, descrito no enunciado, será uma união enumerável de enumeráveis. Seja então m uma função de $\mathbb{N}$ em $\mathbb{N}-\{0\}$, para a qual exista no máximo um elemento do domínio cuja imagem seja igual a 1.

Para $i\in\mathbb{N}$, temos os alfabetos da fita $\Gamma_i=\{\alpha_{i,0},\alpha_{i,1},\ldots,\alpha_{i,m(i)-1}\}$. Para cada um deles, há exatamente $2^{m(i)-1}$ alfabetos de entrada Σ possíveis, visto que $\Sigma\subseteq\Gamma-\{\textvisiblespace\}$. Ou seja, cada um traz consigo uma quantidade finita de alfabetos de entrada.

Para cada alfabeto da fita há também uma quantidade infinita enumerável de conjuntos de estado $Q_n=\{\mathfrak{q}_0,\mathfrak{q}_1,\ldots,\mathfrak{q}_{n-1}\}$, onde $n\in\mathbb{N}-\{0,1\}$. Para uma máquina formada a partir de um deles, a função de transição $\delta: Q\times\Gamma\to Q\times\Gamma\times\{\mathsf{R},\mathsf{L}\}$ dispõe de um domínio e de um contradomínio finitos e, portanto, existe em finitas possibilidades distintas, de acordo com as diferentes imagens que pode assumir para cada elemento do domínio.

Ainda para uma dessas máquinas, as possíveis opções por qual será o estado inicial estão em quantidade de n. Também é igual a n a quantidade de possíveis opções pelo estado de aceitação. E daí é igual a $n-1$ a quantidade relativa ao estado de rejeição.

Com isso, fixado número $n_0\in\mathbb{N}$ qualquer, e com a sendo algum elemento do conjunto $\{0,1,\ldots,n-1\}$, olhemos para o conjunto

$$M(n_0)=\Big\{(Q_{n_0},\Sigma,\Gamma_{i_0},\delta,\mathfrak{q}_a,\mathfrak{q}_{\text{aceita}},\mathfrak{q}_{\text{rejeita}});\ \Sigma\subseteq\Gamma_{i_0}-\{\textvisiblespace\},$$

$$\delta: Q_{n_0}\times\Gamma_{i_0}\to Q_{n_0}\times\Gamma_{i_0}\times\{\mathsf{R},\mathsf{L}\}$$

$$\text{e } \mathfrak{q}_a,\mathfrak{q}_{\text{aceita}},\mathfrak{q}_{\text{rejeita}}\in Q_{n_0},\mathfrak{q}_{\text{aceita}}\neq\mathfrak{q}_{\text{rejeita}}\Big\}.$$

Por tudo o que acabamos de falar, conseguimos afirmar que $M(n_0)$ é um conjunto finito. Seja, então, $M=\bigcup_{n\in\mathbb{N}-\{0,1\}} M(n)$. Trata-se exatamente do conjunto de todas as máquinas obtidas a partir do alfabeto da fita considerado inicialmente. E como é a reunião enumerável de conjuntos enumeráveis, ele é, então, enumerável. ■

Tal teorema nos dá um resultado mais geral. E a partir dele e da observação 4, chegamos a um resultado mais específico.

Corolário 3.6.2. *A classe das máquinas de Turing iguais a $\mathcal{M}^{\mathcal{T}}$ é um conjunto enumerável.* ■

A partir de tais máquinas, obtemos as sequências computáveis. E mesmo que nem todas essas máquinas cheguem a imprimir uma sequência assim, as sequências computáveis, como lembramos antes, existem em quantidade infinita. Logo, o conjunto delas é infinito e enumerável. E daí, pelo argumento da reunião dos números computáveis, chegamos à seguinte conclusão.

Corolário 3.6.3. *O conjunto dos números computáveis é enumerável.* ■

Há um último teorema, ainda, para esta seção, onde resgatamos observações anteriores.

Teorema 3.6.4. *Quanto aos números transcendentes:*

a) *há uma quantidade enumerável deles que são computáveis;*

b) *há uma quantidade não-enumerável deles que não são computáveis.*

Demonstração. Lembremo-nos do que diziam as observações 1 e 2. Ora, o conjunto dos transcendentes é não-enumerável. Mas também é verdade que o conjunto dos números computáveis transcendentes não é vazio (vide o número π).

Se representássemos o conjunto dos transcendentes como a união entre os transcendentes computáveis e os transcendentes não computáveis, então estes dois conjunto ou seriam não-enumeráveis, ou apenas um deles seria enumerável. Mas, como vimos há pouco, os números computáveis existem em quantidade enumerável.

Assim, a primeira opção é excluída. E, sendo um destes dois conjuntos enumerável, tem que ser o dos transcendentes computáveis. Logo, está provado. ■

Juntando as informações que reunimos até este momento, vemos algumas coisas. Todos os números algébricos são computáveis, mas nem todos os números computáveis são algébricos.

E, pelo que já tratamos antes, o número π não é o único transcendente. Não só não é o único, como na verdade ele está acompanhado de infinitos outros números. E mais, o conjunto destes, que é não-enumerável, quando comparado ao conjunto dos algébricos, é descomunal.

Assim, dada a enumerabilidade do conjunto dos computáveis, vemos que “existem muito mais números não-computáveis do que números computáveis”. E também, que “existem muito mais números transcendentes não-computáveis do que transcendentes computáveis”. Mesmo que estes últimos conjuntos sejam todos infinitos.

Nesta seção, é bom fazermos menção também à estratégia utilizada por Turing para mostrar que suas máquinas lhe garantiam uma quantidade enumerável de sequências computáveis. Aproveitaremos algo dela quando formos falar da máquina de Turing universal.

Sejam as nossas máquinas $\mathcal{M}^{\mathcal{T}} = \left(Q^{\mathcal{T}}, \varnothing, \Gamma^{\mathcal{T}}, \delta^{\mathcal{T}}, \mathfrak{q}_0, \mathfrak{q}_{n-2}, \mathfrak{q}_{n-1}\right)$. Uma máquina de Turing dessas computa sequências partindo de um estado inicial $\mathfrak{q}_0$, da leitura dos símbolos escaneados pela cabeça, das operações que serão feitas a cada leitura e da sucessão de estados. E dada sua condição de existência, onde a única cadeia de entrada que recebe é a cadeia vazia, a máquina computa uma única sequência, se é que computa alguma.

Falamos isso pois poderíamos facilmente pensar em uma máquina de Turing que imprimisse apenas dentro de E-quadrados, mesmo que traga a impressão de F--símbolos em sua descrição, as quais nunca acontecerão.

Além disso, podemos pensar em diferentes máquinas de Turing que computem a mesma sequência. Por exemplo, a sequência do número $\left(0, \overline{01}\right)_2$ pode ser computada por uma máquina que coloca um marcador após cada dígito zero e um outro marcador após cada dígito um. Mas a mesma sequência também pode ser computada por uma máquina de Turing que não use marcador algum.

A estratégia pensada é identificar cada máquina de Turing com um número natural. Com isso, consequentemente, estaremos associando cada um desses números naturais também a uma "sequência computável". Às vezes, diversos números serão associados a uma mesma sequência advinda de dado número computável. Às vezes serão associados a uma "sequência computável" que não contenha F-símbolos.

Pois bem, cada sequência computável de um número computável possuirá algum número que a ela é associado. E, ao total, estes números naturais estarão em quantidade infinita e enumerável.

Reformulando, sejam T o conjunto de todas as máquinas de Turing $\mathcal{M}^{\mathcal{T}}$, $S_{\mathcal{M}^{\mathcal{T}}}$ a sequência que uma máquina $\mathcal{M}^{\mathcal{T}}$ computa em seus F-quadrados, e S o conjunto de todas as sequências $S_{\mathcal{M}^{\mathcal{T}}}$. Encontraremos uma bijeção $f : T \to \mathbb{N}'$, sendo $\mathbb{N}' \subset \mathbb{N}$ infinito. E, sendo a função sobrejetora $g : T \to S$, $g\left(\mathcal{M}^{\mathcal{T}}\right) = S_{\mathcal{M}^{\mathcal{T}}}$, temos que a função $g \circ f^{-1} : \mathbb{N}' \to S$ nos diz que as sequências computáveis são enumeráveis.

Caso tenhamos uma máquina $\mathcal{M}^{\mathcal{T}}$ que não imprima em nenhum F-quadrado, nós vamos dizer que $S_{\mathcal{M}^{\mathcal{T}}} = \varepsilon$. Agora, trataremos de explicar o que estará por trás da bijeção f acima comentada.

Podemos descrever $\mathcal{M}^{\mathcal{T}}$ colocando em ordem toda a ação prevista na máquina, a partir do estado inicial, passando por todos os estados, através de cada símbolo escaneável. Com $i, j \in \mathbb{N}$, $0 \leq i \leq n$ e $0 \leq j \leq m$, sejam $\alpha_{i,j}$, $\mathsf{D}_{i,j} \in \{\mathsf{R}, \mathsf{L}\}$ e $\mathfrak{q}_{i,j}$ respectivamente o símbolo a ser impresso, a direção a ser tomada pela cabeça e o estado sucessor quando a máquina está no estado $\mathfrak{q}_i$ e escaneia o símbolo α_j. Abaixo

encontramos uma descrição tal como dissemos para esta máquina.

$$\mathfrak{q}_0\alpha_0\alpha_{0,0}\mathsf{D}_{0,0}\mathfrak{q}_{0,0}\ ;\ \mathfrak{q}_0\alpha_1\alpha_{0,1}\mathsf{D}_{0,1}\mathfrak{q}_{0,1}\ ;\ \ldots\ ;\ \mathfrak{q}_0\alpha_{m-1}\alpha_{0,m-1}\mathsf{D}_{0,m-1}\mathfrak{q}_{0,m-1}\ ;$$

$$\mathfrak{q}_1\alpha_0\alpha_{1,0}\mathsf{D}_{1,0}\mathfrak{q}_{1,0}\ ;\ \ldots\ ;\ \mathfrak{q}_1\alpha_{m-1}\alpha_{1,m-1}\mathsf{D}_{1,m-1}\mathfrak{q}_{1,m-1}\ ;\ \ldots\ ;$$

$$\mathfrak{q}_{n-1}\alpha_0\alpha_{n-1,0}\mathsf{D}_{n-1,0}\mathfrak{q}_{n-1,0}\ ;\ \ldots\ ;\ \mathfrak{q}_{n-1}\alpha_{m-1}\alpha_{n-1,m-1}\mathsf{D}_{n-1,m-1}\mathfrak{q}_{n-1,m-1}$$

Chamaremos esta descrição para nossas máquinas de Turing de **descrição algébrica ordenada de** $\mathcal{M}^{\mathcal{T}}$. E a iremos substituir por um código de letras correspondente. Os estados $\mathfrak{q}_i$ serão substituídos, respectivamente, pela letra D seguida da letra A em quantidade de $i+1$. Ou seja, o estado $\mathfrak{q}_0$ é substituído por DA, o estado $\mathfrak{q}_1$ é substituído por DAA, o $\mathfrak{q}_2$ é substituído por DAAA, e assim por diante.

Os símbolos α_j serão, respectivamente, substituídos pela letra D seguida da letra C em quantidade de j. Ou seja, o símbolo α_0 é substituído por D, o símbolo α_1 é substituído por DC, o α_2 é substituído por DCC, e assim por diante. Desta maneira, não haverá uma futura confusão entre o símbolo α_0 e o estado $\mathfrak{q}_0$ como haveria se substituíssemos o estado inicial também por D simplesmente.

As operações de deslocamento, R e L, continuam sendo representadas assim, uma vez que são indicadas por uma letra cada. E, para separarmos cada uma dessas 5-uplas, estamos utilizando o ";". Mantê-lo-emos entre cada um dos códigos que substituirá tais 5-uplas. A esta escrita, com tais letras seguindo esta ordem, para uma de nossas máquinas, damos o nome de **descrição padrão de** $\mathcal{M}^{\mathcal{T}}$.

Por exemplo, temos aqui abaixo a descrição tabular de uma máquina de Turing que imprime a sequência relativa ao número $(0,\overline{001})_2$. Sejam então, nela, $\alpha_4 = \mathbf{x}$ e $\alpha_5 = \mathbf{y}$.

Observamos que, em verdade, a descrição da atividade da máquina nos estados $\mathfrak{q}_{11}$ e $\mathfrak{q}_{12}$ é indiferente, tal como sempre o é para os estados de aceitação e de rejeição em qualquer situação. Pois as suas operações enquanto estiver sobre eles e a sucessão de estado indicada nunca serão realizadas, uma vez que, no momento em que as máquinas de Turing trocam para qualquer um dos dois, elas terminam todas as suas atividades.

Logo após a descrição tabular, nós colocamos a descrição algébrica ordenada da máquina de Turing então tomada. Está feita de maneira que a organizamos pelos seus estados, desde o inicial até o final em ordem numérica, e, dentro de cada estado, organizamos a descrição pelos símbolos escaneáveis, desde o inicial até o final, também em ordem numérica.

E abaixo da descrição algébrica ordenada, colocamos a descrição padrão da mesma máquina de Turing.

Tabela 38 – Descrição tabular do exemplo para uma máquina $\mathcal{M}^{\mathcal{T}}$

Estado	Símbolo	Operações	Estado final
$\mathfrak{q}_0$	Qualquer	P$\mathfrak{d}$, R	$\mathfrak{q}_1$
$\mathfrak{q}_1$	Qualquer	P$\mathfrak{d}$, R	$\mathfrak{q}_2$
$\mathfrak{q}_2$	Qualquer	P0, R	$\mathfrak{q}_3$
$\mathfrak{q}_3$	Qualquer	P$\mathbf{x}$, L	$\mathfrak{q}_4$
$\mathfrak{q}_4$	β	Pβ, R	$\mathfrak{q}_5$
$\mathfrak{q}_5$	$\mathbf{x}$	P$\textvisiblespace$, R	$\mathfrak{q}_6$
	Não $\mathbf{x}$	P$\textvisiblespace$, R	$\mathfrak{q}_9$
$\mathfrak{q}_6$	Qualquer	P0, R	$\mathfrak{q}_7$
$\mathfrak{q}_7$	Qualquer	P$\mathbf{y}$, L	$\mathfrak{q}_8$
$\mathfrak{q}_8$	β	Pβ, R	$\mathfrak{q}_5$
$\mathfrak{q}_9$	Qualquer	P1, R	$\mathfrak{q}_{10}$
$\mathfrak{q}_{10}$	Qualquer	P$\textvisiblespace$, R	$\mathfrak{q}_2$
$\mathfrak{q}_{11}$	Qualquer	P1, R	$\mathfrak{q}_1$
$\mathfrak{q}_{12}$	Qualquer	P1, R	$\mathfrak{q}_1$

Fonte: Autor (2024)

$$\mathfrak{q}_0\alpha_0\alpha_3\mathsf{R}\mathfrak{q}_1;\ \mathfrak{q}_0\alpha_1\alpha_3\mathsf{R}\mathfrak{q}_1;\ \mathfrak{q}_0\alpha_2\alpha_3\mathsf{R}\mathfrak{q}_1;\ \mathfrak{q}_0\alpha_3\alpha_3\mathsf{R}\mathfrak{q}_1;\ \mathfrak{q}_0\alpha_4\alpha_3\mathsf{R}\mathfrak{q}_1;\ \mathfrak{q}_0\alpha_5\alpha_3\mathsf{R}\mathfrak{q}_1;$$

$$\vdots$$

$$\mathfrak{q}_4\alpha_0\alpha_0\mathsf{R}\mathfrak{q}_5;\ \mathfrak{q}_4\alpha_1\alpha_1\mathsf{R}\mathfrak{q}_5;\ \mathfrak{q}_4\alpha_2\alpha_2\mathsf{R}\mathfrak{q}_5;\ \mathfrak{q}_4\alpha_3\alpha_3\mathsf{R}\mathfrak{q}_5;\ \mathfrak{q}_4\alpha_4\alpha_4\mathsf{R}\mathfrak{q}_5;\ \mathfrak{q}_4\alpha_5\alpha_5\mathsf{R}\mathfrak{q}_5;$$

$$\vdots$$

$$\mathfrak{q}_{12}\alpha_3\alpha_2\mathsf{R}\mathfrak{q}_1;\ \mathfrak{q}_{12}\alpha_4\alpha_2\mathsf{R}\mathfrak{q}_1;\ \mathfrak{q}_{12}\alpha_5\alpha_2\mathsf{R}\mathfrak{q}_1;$$

DADDCCCRDAA;DADCDCCCRDAA;DADCCDCCCRDAA;

DADCCCDCCCRDAA;DADCCCCDCCCRDAA;DADCCCCCDCCCRDAA;

$$\vdots$$

DAAAAADDRDAAAAAA;DAAAAADCDCRDAAAAAA;

DAAAAADCCDCCRDAAAAAA;DAAAAADCCCDCCCRDAAAAAA;

DAAAAADCCCCDCCCCRDAAAAAA;

DAAAAADCCCCCDCCCCCRDAAAAAA;

$$\vdots$$

DAAAAAAAAAAAADCCCDCCRDAA;

DAAAAAAAAAAAAADCCCCDCCRDAA;

DAAAAAAAAAAAAADCCCCCDCCRDAA;

Seja $\Sigma_3 = \{\mathsf{A}, \mathsf{C}, \mathsf{D}, \mathsf{R}, \mathsf{L}, ;\}$. Como podemos ver, cada máquina $\mathcal{M}^{\mathcal{T}}$, por ter uma descrição algébrica ordenada única, é traduzida em um elemento de Σ_3^* único. Chamaremos tal elemento de $w_{\mathcal{M}^{\mathcal{T}}}$. E ainda, cada elemento de Σ_3^*, pelo procedimento inverso, quando possível, é convertível em uma descrição ordenada algébrica única de uma máquina $\mathcal{M}^{\mathcal{T}}$, e, consequentemente, para $\mathcal{M}^{\mathcal{T}}$.

Com isso, seja $W \subset \Sigma_3^*$, $W = \left\{w_{\mathcal{M}^{\mathcal{T}}} \in \Sigma_3^*;\ \text{para } \mathcal{M}^{\mathcal{T}} \in T\right\}$. Temos, então, uma bijeção $f_1 : T \to W$, $f_1\left(\mathcal{M}^{\mathcal{T}}\right) = w_{\mathcal{M}^{\mathcal{T}}}$.

A partir disso, fazemos um último e simples procedimento de substituição. Para os elementos $w_{\mathcal{M}^{\mathcal{T}}}$, cada letra “A” é substituída pelo algarismo “1”, cada letra “C” é substituída pelo algarismo “2”, cada “D” é substituída por “3”, cada “R” é substituída por “4”, cada “L” por “5” e cada “;” por “6”.

Sendo assim, interpretemos cada cadeia desses dígitos, obtida como descrito acima, como um número natural. Temos que cada $w_{\mathcal{M}^{\mathcal{T}}}$ possui, por este método, um correspondente $k_{\mathcal{M}^{\mathcal{T}}} \in \mathbb{N}$ único.

Utilizando o exemplo que tomamos acima, da máquina que computa a sequência de $\left(0, \overline{001}\right)_2$, o seu número natural correspondente está abaixo.

$$313322243116313232224311631322322243116313222322243116$$

$$31322223222431163132222232224311631$$

$$\vdots$$

$$3111113343111111631111132324311111163111113223224311111116$$

$$311111322232224311111163111113222232222431111116$$

$$31111132222232222431111116$$

$$\vdots$$

$$3111111111111132223224311631111111111111132222322431163$$

$$311111111111113222223224311631$$

E, de modo análogo ao argumento anterior, é visível que cada elemento do conjunto dos naturais, quando possível, é convertível, pelo procedimento inverso, em uma cadeia $w_{\mathcal{M}^{\mathcal{T}}}$ única. Assim, seja $\mathbb{N}' = \{k_{\mathcal{M}^{\mathcal{T}}} \in \mathbb{N};\ \text{para } w_{\mathcal{M}^{\mathcal{T}}} \in W\}$. Temos, com isso, uma bijeção $f_2 : W \to \mathbb{N}'$, $f_2\left(w_{\mathcal{M}^{\mathcal{T}}}\right) = k_{\mathcal{M}^{\mathcal{T}}}$.

Por fim, tomamos $f = f_2 \circ f_1$. Temos, portanto, a bijeção $f : T \to \mathbb{N}'$, $f\left(\mathcal{M}^{\mathcal{T}}\right) = k_{\mathcal{M}^{\mathcal{T}}}$, onde $\mathbb{N}' \subset \mathbb{N}$.

Esta é, então, outra maneira de provar a enumerabilidade das sequências computáveis. E, consequentemente, dos números computáveis.

Recordemos a proposição 3.2.3, que nos diz que todo número racional é computável. Ora, entre zero e um há infinitos racionais. Assim, já vemos que há infinitas sequências computáveis de números racionais. Consequentemente, há infinitas sequências computáveis.

E como para cada sequência computável há algum número natural associado a ela, segue que tal conjunto de naturais também é infinito. Isto é, $\mathbb{N}'$ é de fato um conjunto infinito.

4 ALGUNS TIPOS DE MÁQUINAS DE TURING

Neste capítulo falaremos de duas outras versões do conceito de máquinas de Turing. Também definidas algebricamente, elas expandem a definição anterior. E, por último, finalmente trataremos da máquina de Turing universal.

4.1 MÁQUINA DE TURING MULTIFITA

A primeira dessas outras versões para máquinas de Turing é pensada como sendo uma máquina com mais fitas. A segunda é pensada como sendo uma máquina com várias possibilidades de transição para uma mesma configuração. Chamamos essas outras versões de **variantes** do modelo da máquina de Turing.

Definição 9. Dizemos que um exemplar de máquina de Turing variante e um exemplar de máquina de Turing original (definição algébrica) são **equivalentes** quando as duas reconhecem a mesma linguagem.

A primeira variante apresentada é a chamada máquina de Turing **multifita**, definida pela 7-upla $(Q, \Sigma, \Gamma, \delta, \mathfrak{q}_0, \mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}})$, com as mesmas condições que uma máquina de Turing como já vínhamos vendo, exceto que $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{\mathsf{R}, \mathsf{L}\}^k$, para $k \geq 2$ natural.

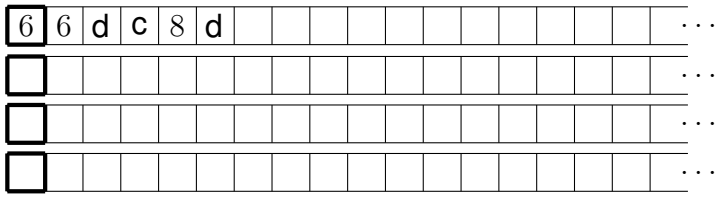
Tal máquina vai ter k fitas e começará com uma cadeia de entrada na primeira delas, estando as demais em branco. Ela igualmente inicia no estado $\mathfrak{q}_0$. Para cada transição, ela lê um símbolo de cada fita, passa para algum estado de Q, imprime um símbolo no quadrado escaneado de cada uma das fitas, e se move para uma direção determinada dentro de cada uma delas. A máquina de Turing multifita funciona, portanto, como se tivesse k cabeças, interdependentes, que agem ao mesmo tempo.

Exemplo 7. Vejamos como funciona uma máquina $\mathcal{M}$ cujo alfabeto da fita é $\Gamma = \{\text{␣}, 6, 8, 9, \mathsf{c}, \mathsf{d}\}$, o alfabeto de entrada é $\Sigma = \Gamma - \{\text{␣}\}$, o conjunto de estados é $Q = \{\mathfrak{q}_0, \mathfrak{q}_1, \mathfrak{q}_2, \mathfrak{q}_3, \mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}}\}$, o estado inicial é $\mathfrak{q}_0$ e tem uma função de transição é $\delta : Q \times \Gamma^4 \to Q \times \Gamma^4 \times \{\mathsf{R}, \mathsf{L}\}^4$. Daremos especificações desta função ao longo do exemplo. Trata-se de uma máquina com quatro fitas.

Tomaremos como entrada, a cadeia $66\mathsf{dc}8\mathsf{d}$. A máquina inicia escaneando o primeiro quadrado de cada uma das fitas simultaneamente, com o estado $\mathfrak{q}_0$.

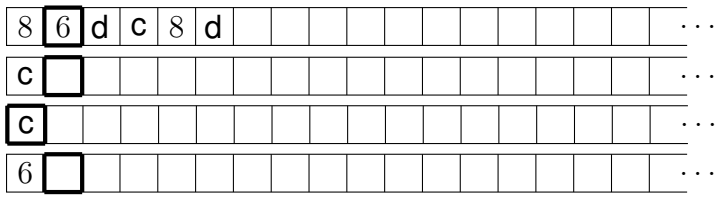
Suponhamos que $\delta(\mathfrak{q}_0, 6, \text{␣}, \text{␣}, \text{␣}) = (\mathfrak{q}_0, 8, \mathsf{c}, \mathsf{c}, 6, \mathsf{R}, \mathsf{R}, \mathsf{L}, \mathsf{R})$. Todos os próximos quadrados escaneados serão o seguinte da direita, exceto o da terceira fita. Afinal a sua cabeça iria para a esquerda mas esbarra com o início da respectiva fita.

Figura 19 – Computação de exemplo de máquina de Turing multifita (parte 1)



Fonte: Autor (2024)

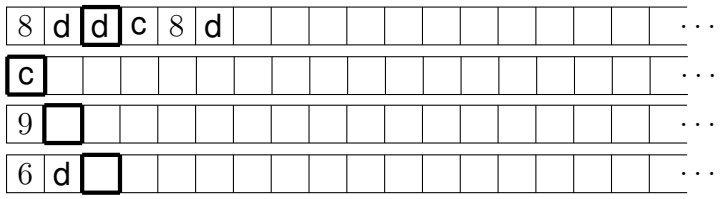
Figura 20 – Computação de exemplo de máquina de Turing multifita (parte 2)



Fonte: Autor (2024)

A máquina permanece no estado $\mathfrak{q}_0$, e continua escaneando um símbolo 6 na primeira fita. Na verdade, todos os símbolos escaneados são iguais aos anteriores, exceto um. O terceiro já não é $\sqcup$, mas sim c. Então o próximo argumento do domínio de δ já não é mais o mesmo. E, na nossa função de transição, temos que $\delta(\mathfrak{q}_0, 6, \sqcup, \mathsf{c}, \sqcup) = (\mathfrak{q}_2, \mathsf{d}, \sqcup, 9, \mathsf{d}, \mathsf{R}, \mathsf{L}, \mathsf{R}, \mathsf{R})$. Então, o próximo estágio da máquina se encontra representado abaixo.

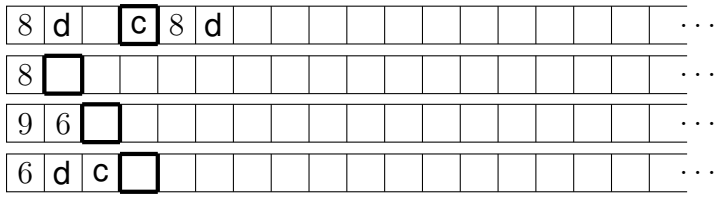
Figura 21 – Computação de exemplo de máquina de Turing multifita (parte 3)



Fonte: Autor (2024)

Depois disso, temos $\delta(\mathfrak{q}_2, \mathsf{d}, \mathsf{c}, \sqcup, \sqcup) = (\mathfrak{q}_1, \sqcup, 8, 6, \mathsf{c}, \mathsf{R}, \mathsf{R}, \mathsf{R}, \mathsf{R})$. Com isso, a máquina fica como na representação abaixo.

Figura 22 – Computação de exemplo de máquina de Turing multifita (parte 4)



Fonte: Autor (2024)

Em seguida, tendo a nossa função de transição sendo tal que $\delta(\mathfrak{q}_1, \mathsf{c}, \sqcup, \sqcup, \sqcup) =$

$(\mathfrak{q}_2, 9, 9, \mathsf{d}, \mathsf{d}, \mathsf{L}, \mathsf{R}, \mathsf{L}, \mathsf{R})$, a máquina fica como a seguir.

Figura 23 – Computação de exemplo de máquina de Turing multifita (parte 5)

Fonte: Autor (2024)

E assim a máquina prossegue. Ela pode entrar em loop. Pode chegar ao estado de aceitação. Ou pode chegar ao estado de rejeição. A opção dentre essas que se realizará depende de como é definida a função de transição. □

Usar uma máquina dessas pode ser conveniente em certas situações. E o incrível é que ela sempre é equivalente a alguma máquina de Turing "monofita". E é isso que iremos provar abaixo.

Teorema 4.1.1. *Toda máquina de Turing multifita é equivalente a alguma máquina de Turing de fita única.*

Demonstração. Seja $\mathcal{S} = (Q, \Sigma, \Gamma, \delta, \mathfrak{q}_0, \mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}})$ uma máquina multifita, onde $\Gamma = \{\alpha_0, \alpha_1, \ldots, \alpha_{m-1}\}$, $Q = \{\mathfrak{q}_0, \mathfrak{q}_1, \ldots, \mathfrak{q}_{n-1}\}$ e $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{\mathsf{R}, \mathsf{L}\}^k$.

Tomaremos cada símbolo do alfabeto da fita e, a partir de cada um, criaremos um novo símbolo com um ponto escrito acima do original. Serão $\alpha_m = \dot{\alpha}_0$, $\alpha_{m+1} = \dot{\alpha}_1$, ..., $\alpha_{2m-1} = \dot{\alpha}_{m-1}$. E sejam também, digamos, $\alpha_{2m} = \#$, $\alpha_{2m+1} = \$$, $\alpha_{2m+2} = ə$, $\alpha_{2m+3} = @$ e $\alpha_{2m+4} = \%$, sendo estes últimos símbolos tais que originalmente não estivessem em Γ.

Sejam, então, Q' um conjunto de estados tal que $Q \subsetneq Q'$, e um alfabeto $\Gamma' = \{\alpha_0, \alpha_1, \ldots, \alpha_{2m+4}\}$. Tomaremos a função $\delta' : Q' \times \Gamma' \to Q' \times \Gamma' \times \{\mathsf{R}, \mathsf{L}\}$, a qual descreveremos a seguir. E também tomaremos a máquina $\mathcal{M} = (Q', \Sigma, \Gamma', \delta', \mathfrak{q}'_0, \mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}})$.

A estratégia para provar tal equivalência se dá criando espaços delimitados lado a lado na única fita onde seriam reproduzidos os símbolos e a computação da máquina multifita. O primeiro espaço é o dos quadrados onde já está a cadeia de entrada. Os demais iniciam vazios tal como as fitas posteriores da máquina multifita também iniciariam.

Dividiremos esta demonstração em quatro partes: **A)** a criação dos espaços para cada fita, **B)** a descrição do mecanismo para reconhecer e operar em cima de cada espaço devidamente, **C)** a descrição do mecanismo de aumento de um espaço quando este ultrapassará sua borda, e **D)** a descrição do mecanismo de aceitação ou de rejeição em imitação à maquina multifita correspondente.

Parte A. Abaixo colocamos alguns dos primeiros passos de $\mathcal{M}$. Ela vai sobrepontear o primeiro símbolo da cadeia de entrada. E após percorrer a entrada sem modificar os demais símbolos, irá imprimir os espaços correspondentes às outras fitas da máquina $\mathcal{S}$, que iniciam em branco.

Para $i = 0, 1, \ldots, m-1$, temos:
$\delta'(\mathfrak{q}'_0, \alpha_i) = (\mathfrak{q}'_1, \alpha_{m+i}, \mathsf{L})$;
$\delta'(\mathfrak{q}'_1, \alpha) = (\mathfrak{q}'_1, \alpha, \mathsf{R})$ se $\alpha \neq \text{␣}$;
$\delta'(\mathfrak{q}'_1, \text{␣}) = (\mathfrak{q}'_2, \text{␣}, \mathsf{L})$;
$\delta'(\mathfrak{q}'_4, \alpha) = (\mathfrak{q}'_4, \alpha, \mathsf{R})$ se $\alpha \neq \text{␣}$;
$\delta'(\mathfrak{q}'_4, \text{␣}) = (\mathfrak{q}'_5, \#, \mathsf{R})$;
$\delta'(\mathfrak{q}'_5, \text{␣}) = (\mathfrak{q}'_6, \$, \mathsf{R})$;
$\delta'(\mathfrak{q}'_6, \text{␣}) = (\mathfrak{q}'_7, \dot{\text{␣}}, \mathsf{R})$;
$\delta'(\mathfrak{q}'_7, \text{␣}) = (\mathfrak{q}'_8, \#, \mathsf{R})$;
$\delta'(\mathfrak{q}'_8, \text{␣}) = (\mathfrak{q}'_9, \$, \mathsf{R})$.

E seguimos o padrão das três últimas imagens até $\delta'(\mathfrak{q}'_{3k}, \text{␣}) = \left(\mathfrak{q}'_{3k+1}, \dot{\text{␣}}, \mathsf{R}\right)$, seguida pela imagem $\delta'\left(\mathfrak{q}'_{3k+1}, \text{␣}\right) = \left(\mathfrak{q}'_{3k+2}, \#, \mathsf{R}\right)$ e pela $\delta'\left(\mathfrak{q}'_{3k+2}, \text{␣}\right) = \left(\mathfrak{q}'_{3(k+1)}, \text{ə}, \mathsf{L}\right)$.

Estamos marcando com o ponto acima, em $\mathcal{M}$, todos os símbolos escaneados das cadeias presentes em cada fita de $\mathcal{S}$. Separaremos, na única fita de $\mathcal{M}$, as cadeias entre si pelo símbolo $\#$ à direita e $\$$ à esquerda. O início dessa sequência de cadeias está delimitado por ə, e o fim também.

A máquina segue com outras transições, mas, por agora, iremos olhar mais afastados da definição de $\mathcal{M}$. A partir daqui até o restante da demonstração, definiremos alguns estados e m-funções de modo que eles possam sobressaltar as convenções de Sipser. Pois, como já provamos, tais elementos poderão ser trocados por estados que obedeçam as convenções.

Mas antes, comecemos com a simplificação da atividade da máquina para um conjunto de estados. Diremos, informalmente, que o trabalho de nossa máquina a partir do estado $\mathfrak{q}'_2$ é descrito pelo pela tabela abaixo.

Tabela 39 – Descrição de $\mathcal{M}$ a partir do estado não convencional $\mathfrak{q}'_2$

Estado		Símbolo	Operações	Estado final
$\mathfrak{q}'_2$	$\left\{\right.$	α_i	R, R, Pα_i, L, L, P$_{\text{␣}}$, L	$\mathfrak{q}'_2$
		β, não α_i	R, R, Pβ, L, L, P$_{\text{␣}}$	$\mathfrak{q}'_3$
$\mathfrak{q}'_3$		Qualquer	Pə, R, P$, R	$\mathfrak{q}'_4$

Fonte: Autor (2024)

É com tal estado que movimentamos a cadeia de entrada duas unidades para frente. Imprimimos no início da fita o ə e o $\#$. Quando chegamos a $\mathfrak{q}'_4$, começaremos a imprimir os símbolos relativos às cadeias das demais fitas de $\mathcal{S}$, como também os símbolos que as separam.

Parte B. Definimos, neste momento, uma notação para sequência de argumentos em uma m-função. Para $a \in \mathbb{N}$ positivo, denotamos $(\sigma_a)_x = (\gamma_1, \ldots, \gamma_a)$, onde $\gamma_1, \ldots, \gamma_a \in \Gamma$ e x pode ser qualquer coisa, para marcar a sequência e diferenciá-la de outras de mesmo tamanho. E definimos que $(\sigma_0)_x$ significa ausência de uma sequência. Por exemplo, para uma m-função $\mathfrak{h}$, temos $\mathfrak{h}\left(\mathfrak{q}_2, \mathfrak{q}'_4, (\sigma_3)_x, \alpha_{m+5}\right) = \mathfrak{h}(\mathfrak{q}_2, \mathfrak{q}'_4, \gamma_1, \gamma_2, \gamma_3, \alpha_{m+5})$, e também $\mathfrak{h}\left(\mathfrak{q}_2, \mathfrak{q}'_4, (\sigma_0)_x, \alpha_{m+5}\right) = \mathfrak{h}\left(\mathfrak{q}_2, \mathfrak{q}'_4, \alpha_{m+5}\right)$.

Nesta parte definiremos três tipos de m-funções, a partir das quais será feito o trabalho principal da máquina.

O primeiro tipo é o das m-funções $\mathfrak{fp}$, cuja tarefa estabelecida é de encontrar os símbolos sobreponteados. Eles indicam quais seriam os quadrados escaneados na máquina multifita. Ao passo que são encontrados, seus estados vão passando adiante a informação dos k símbolos escaneados a cada vez.

O segundo tipo é o das m-funções $\mathfrak{gw}$, cuja tarefa estabelecida é a de operar em cada um dos k espaços do modo como a máquina multifita faria. Ou seja, é impresso no lugar certo um novo símbolo e fica garantido o deslocamento para direita ou esquerda.

O terceiro tipo é o das m-funções $\mathfrak{dp}$, cuja tarefa estabelecida é a de deixar o próximo símbolo sobreponteado em cada espaço após feitas as respectivas operações, para que, com as m-funções $\mathfrak{fp}$, o ciclo possa continuar. Isto indica quais seriam os próximos quadrados escaneados em cada fita da máquina $\mathcal{S}$.

Seja, então, $b \in \mathbb{N}$, com $1 \leq b \leq k-1$. Introduzimos as m-funções $\mathfrak{fp}$, de *find point*, que têm um estado e $(k+b)$ símbolos como argumentos.

Tabela 40 – Descrição de $\mathcal{M}$ a partir das imagens das m-funções *find point* de $(k+b+1)$ argumentos

Estado		Símbolo	Op.	Estado final
$\mathfrak{fp}\left(\mathfrak{C}, (\sigma_k)_z, (\sigma_b)_y\right)$	$\left\{\right.$	$\#$	R	$\mathfrak{fp}_1\left(\mathfrak{C}, (\sigma_k)_z, (\sigma_b)_y\right)$
		Não $\#$	R	$\mathfrak{fp}\left(\mathfrak{C}, (\sigma_k)_z, (\sigma_b)_y\right)$
$\mathfrak{fp}_1\left(\mathfrak{C}, (\sigma_k)_z, (\sigma_b)_y\right)$	$\left\{\right.$	α_{m+i}		$\mathfrak{gw}\left(\mathfrak{C}, (\sigma_k)_z, (\sigma_b)_y\right)$
		Não α_{m+i}	R	$\mathfrak{fp}_1\left(\mathfrak{C}, (\sigma_k)_z, (\sigma_b)_y\right)$

Fonte: Autor (2024)

A partir do estado $\mathfrak{fp}\left(\mathfrak{C}, (\sigma_k)_z, (\sigma_b)_y\right)$, a máquina procura, na próxima cadeia entre dois símbolos $\#$, o próximo símbolo sobreponteado da fita, e então mudará para o estado $\mathfrak{gw}\left(\mathfrak{C}, (\sigma_k)_z, (\sigma_b)_y\right)$.

E ainda, definimos a atividade a partir da imagem da função $\mathfrak{fp}$ com $(k+1)$ argumentos. Encontra-se na tabela 41.

Quanto a $\mathfrak{gw}$, de *get to work*, esta notação está atrelada a k m-funções que aqui definimos. A máquina $\mathcal{S}$ tem um símbolo escaneado para cada uma das k fitas. Chamaremos o símbolo escaneado da sua primeira fita de β_1, o da sua segunda fita de β_2, até o símbolo escaneado da sua k-ésima fita, que será β_k.

Tabela 41 – Descrição de $\mathcal{M}$ a partir das imagens da m-função *find point* de $(k+1)$ argumentos

Estado		Símbolo	Operações	Estado final
$\mathfrak{fp}\left(\mathfrak{B},(\sigma_k)_z\right)$	$\left\{\right.$	ə	R	$\mathfrak{fp}_1\left(\mathfrak{B},(\sigma_k)_z\right)$
		Não ə	L	$\mathfrak{fp}\left(\mathfrak{B},(\sigma_k)_z\right)$
$\mathfrak{fp}_1\left(\mathfrak{B},(\sigma_k)_z\right)$	$\left\{\right.$	α_{m+i}		$\mathfrak{gw}\left(\mathfrak{B},(\sigma_k)_z\right)$
		Não α_{m+i}	R	$\mathfrak{fp}_1\left(\mathfrak{B},(\sigma_k)_z\right)$

Fonte: Autor (2024)

Para $r=1,2,\ldots,k$, em $\mathcal{M}$, quando um símbolo β_r for trocado por sua versão sobreponteada, chamaremos esta versão de β_{-r}. Isto é, $\beta_{-r}=\dot{\beta}_r$. Para definirmos o comportamento da máquina a partir dos estados advindos das m-funções $\mathfrak{gw}$, observemos que, para $i_r\in\{0,1,\ldots,m-1\}$ e $\mathsf{X}_r\in\{\mathsf{R},\mathsf{L}\}$, temos, para $q\in Q$ e algum $q'\in Q$, que

$$\delta\left(q,\beta_1,\beta_2,\ldots,\beta_k\right)=\left(q',\alpha_{i_1},\alpha_{i_2},\ldots,\alpha_{i_k},\mathsf{X}_1,\mathsf{X}_2,\ldots,\mathsf{X}_k\right)=$$

$$=\left(\delta_1\left(q,\beta_1,\beta_2,\ldots,\beta_k\right),\delta_2\left(q,\beta_1,\beta_2,\ldots,\beta_k\right),\ldots,\delta_{2k+1}\left(q,\beta_1,\beta_2,\ldots,\beta_k\right)\right).$$

Dado tudo isso, sendo $\mathfrak{B}\in Q-\{\mathfrak{q}_{\text{aceita}},\mathfrak{q}_{\text{rejeita}}\}$ tratemos de $\mathfrak{gw}$. A partir dos estados advindos da m-função $\mathfrak{gw}$, a máquina $\mathcal{M}$ vai fazendo, ao longo dos símbolos sobreponteados de sua fita, as operações que a máquina $\mathcal{S}$ faria ao longo das suas, nos seus quadrados escaneados.

Tabela 42 – Descrição de $\mathcal{M}$ a partir das imagens das m-funções *get to work* (parte 1)

Estado	Símb.	Op.	Estado final
$\mathfrak{gw}\left(\mathfrak{B},(\sigma_k)_z\right)$	Qualquer	$\mathsf{P}\alpha_{i_1}$, X_1	$\mathfrak{dp}\left(\mathfrak{B},(\sigma_k)_z\right)$
$\mathfrak{gw}\left(\mathfrak{B},(\sigma_k)_z,(\sigma_b)_y\right)$	Qualquer	$\mathsf{P}\alpha_{i_{b+1}}$, X_{b+1}	$\mathfrak{dp}\left(\mathfrak{B},(\sigma_k)_z,(\sigma_b)_y\right)$

Fonte: Autor (2024)

Como dissemos, a descrição acima está feita para $\mathfrak{B}\in Q-\{\mathfrak{q}_{\text{aceita}},\mathfrak{q}_{\text{rejeita}}\}$. Mas ainda restam os outros estados de Q'. E ainda, se o estado a que $\mathcal{S}$ chegasse fosse o estado de aceitação ou o de rejeição, temos de fazer com que ele também seja estado de $\mathcal{M}$. Sendo $\mathfrak{E}\in(Q'-Q)\cup\{\mathfrak{q}_{\text{aceita}},\mathfrak{q}_{\text{rejeita}}\}$.

Tabela 43 – Descrição de $\mathcal{M}$ a partir das imagens das m-funções *get to work* (parte 2)

Estado	Símbolo	Operações	Estado final
$\mathfrak{gw}\left(\mathfrak{E},(\sigma_k)_z\right)$	Qualquer		$\mathfrak{E}$

Fonte: Autor (2024)

Poderíamos descrever também a atividade de $\mathcal{M}$ sobre as imagens das versões de $\mathfrak{gw}$ com mais argumentos a partir dos estados restantes de Q', mas não será necessário. Elas não farão diferença na atividade da máquina, então sua descrição fica livre.

Em verdade, pelo modo como vão ser colocados tais estados dentro da máquina, a partir deles a cabeça sempre escaneará exatamente os β_{-r}, afinal as m-funções $\mathfrak{fp}$ envolvidas as conduzem a eles. Apenas colocamos, na tabela, as mesmas operações da máquina no estado como se dando para qualquer símbolo a fim de completar a descrição.

Após fazer as operações correspondentes em um símbolo sobreponteado, a máquina passa para o estado advindo da m-função $\mathfrak{dp}$, com os mesmos argumentos que tinha no estado recente.

Apresentamos, então, as m-funções $\mathfrak{dp}$, de *draw a point*, a seguir. Com a tabela respectiva, não estamos descrevendo $\mathcal{M}$ apenas sobre as imagens de um par de m-funções, mas sim de $(k-1)$ pares de m-funções. Cada par com $(k+b)$ argumentos.

Tabela 44 – Descrição de $\mathcal{M}$ a partir das imagens das m-funções *draw a point* de $(k+b)$ argumentos

Estado	Símbolo	Op.	Estado final
$\mathfrak{dp}(\mathfrak{C},(\sigma_k)_z,(\sigma_{b-1})_y)$	α_i	P$_{\alpha_{m+i}}$	$\mathfrak{fp}(\mathfrak{C},(\sigma_k)_z,(\sigma_{b-1})_y,\alpha_i)$
	$\#$		$\mathfrak{bpem}(\mathfrak{fp}(\mathfrak{C},(\sigma_k)_z,(\sigma_{b-1})_y))$
	Não α_i e não $\#$	R	$\mathfrak{dp}_1(\mathfrak{C},(\sigma_k)_z,(\sigma_{b-1})_y)$
$\mathfrak{dp}_1(\mathfrak{C},(\sigma_k)_z,(\sigma_{b-1})_y)$	α_i	P$_{\alpha_{m+i}}$	$\mathfrak{fp}(\mathfrak{C},(\sigma_k)_z,(\sigma_{b-1})_y,\alpha_i)$
	Não α_i		$\mathfrak{q}_0$

Fonte: Autor (2024)

Em tal estado $\mathfrak{dp}\,(\mathfrak{C},(\sigma_k)_z,(\sigma_b)_y)$, a máquina imprime, no lugar de um símbolo α_i do alfabeto da fita de $\mathcal{S}$, a versão desse símbolo com o ponto em cima, e muda para o estado $\mathfrak{fp}\,(\mathfrak{C},(\sigma_k)_z,(\sigma_b)_y,\alpha_i)$. Se isso não acontecer, é porque a máquina vai ter escaneado, realmente, ou $\$$, ou $\#$.

Se $\$$, ela vai para a direita, escaneia um símbolo α_i e imprime a versão sobreponteada dele. Em seguida, muda para o estado $\mathfrak{fp}\,(\mathfrak{C},(\sigma_k)_z,(\sigma_b)_y,\alpha_i)$. Se $\#$, ela então seguirá para estados sobre os quais comentaremos mais à frente.

Também definimos, na tabela 45, a atividade de $\mathcal{M}$ a partir das imagens da m-função $\mathfrak{dp}$ de $2k$ argumentos. A partir do estado $\mathfrak{dp}\,(\mathfrak{C},(\sigma_k)_z,(\sigma_{k-1})_y)$ a máquina imprime o sobreponteado na cadeia relativa à k-ésima fita de $\mathcal{S}$. E muda para o estado $\mathfrak{fp}\,(\delta_1(\mathfrak{C},(\sigma_k)_z),(\sigma_{k-1})_y,\alpha_i)$. O estado $\mathfrak{q}_0$ destas duas últimas tabelas só está ali para completar a descrição. Pois, nos estados advindos das m-funções $\mathfrak{dp}_1$, a cabeça sempre escaneará um elemento de Γ.

Após a máquina sobrepontear o símbolo correspondente à k-ésima fita de $\mathcal{S}$, ela volta ao começo da fita e vai, então, àquele que é agora o novo primeiro símbolo sobreponteado. E o padrão recomeça, sendo o antigo estado $\mathfrak{B}$, como argumento de $\mathfrak{gw}$, substituído pelo estado em que a máquina $\mathcal{S}$ estaria após escanear todos os símbolos β_r anteriores durante próprio o estado $\mathfrak{B}$. É para isso que vamos carregando, como argumento, símbolo após símbolo enquanto percorremos da primeira até a última das k cadeias separadas.

Tabela 45 – Descrição de $\mathcal{M}$ a partir das imagens da m-função *draw a point* de $2k$ argumentos

Estado	Símbolo	Op.	Estado final
$\mathfrak{dp}(\mathfrak{C},(\sigma_k)_z,(\sigma_{k-1})_y)$	α_i	Pα_{m+i}	$\mathfrak{fp}(\delta_1(\mathfrak{C},(\sigma_k)_z),(\sigma_{k-1})_y,\alpha_i)$
	$\#$		$\mathfrak{bpem}(\mathfrak{fp}(\mathfrak{C},(\sigma_k)_z,(\sigma_{k-1})_y))$
	Não α_i e não $\#$	R	$\mathfrak{dp}_1(\mathfrak{C},(\sigma_k)_z,(\sigma_{k-1})_y)$
$\mathfrak{dp}_1(\mathfrak{C},(\sigma_k)_z,(\sigma_{k-1})_y)$	α_i	P α_{m+i}	$\mathfrak{fp}(\delta_1(\mathfrak{C},(\sigma_k)_z),(\sigma_{k-1})_y,\alpha_i)$
	Não α_i		$\mathfrak{q}_0$

Fonte: Autor (2024)

Parte C. Introduzimos, também, as m-funções $\mathfrak{bpem}$, de *begin printing at the end and mark*, e $\mathfrak{pem}$, de *print at the end and mark*, inspiradas na m-função *print at the end*. Elas e as demais m-funções a seguir são usadas para quando, durante a troca de símbolos sobreponteados, a cabeça se encontrar acima de $\#$. Isso significa que, em $\mathcal{S}$, a máquina estaria expandindo sua cadeia impressa para além do final de uma fita.

E na máquina $\mathcal{M}$, não queremos que isso dê problema nas cadeias que estão lado a lado, separadas por $\#$ e $\$$. Daí, se a cabeça iria sobrepontear um símbolo, mas parou em $\#$, seguimos o procedimento de mover para depois do atual fim da fita todas as cadeias que estão à direita da cadeia atual, junto com seus delimitadores, deixando o antigo espaço delas em branco.

Explicando em que consistem os passos descritos abaixo, basicamente, iremos copiar todos os símbolos à direita do $\#$, onde a cabeça a sobrepontear se encontrou, para o fim da fita, em ordem. Para isso, vamos substituindo-os por $@$, enquanto marcamos o novo fim da fita com $\%$. Quando vamos copiar o ə, chegamos ao último símbolo com o qual repetimos esse padrão. Ele substitui $\%$ no fim da fita. Substituímos o primeiro dos $@$ por $\dot{\sqcup}$ e apagamos os demais. Partimos, então, para a cadeia relativa à próxima fita de $\mathcal{S}$, ou, se já estávamos na última, voltamos à primeira.

Dito isso, vejamos na tabela a atividade de $\mathcal{M}$ a partir das imagens das m-funções antes citadas. Com o primeiro desses estados, $\mathfrak{bpem}(\mathfrak{C})$, são escaneados indiferentemente os símbolos. Mas o real papel dele na máquina será fazê-la substituir o símbolo $\#$, que é o símbolo que ela imediatamente encontrará, pelo símbolo $@$.

Tabela 46 – Descrição de $\mathcal{M}$ a partir das imagens das m-funções $\mathfrak{bpem}$ e $\mathfrak{pem}$

Estado	Símbolo	Operações	Estado final
$\mathfrak{bpem}(\mathfrak{C})$	Qualquer	P$@$	$\mathfrak{pem}(\mathfrak{C})$
$\mathfrak{pem}(\mathfrak{C})$	ə	R, P$\#$ R, P$\%$	$\mathfrak{f}(\mathfrak{faf}(\mathfrak{C}),@)$
	Não ə	R	$\mathfrak{pem}(\mathfrak{C})$

Fonte: Autor (2024)

O segundo desses estados, $\mathfrak{pem}(\mathfrak{C})$, a fará deixar impresso, imediatamente à direita do fim da fita, o símbolo $\#$, imprimir $\%$ logo à direita deste, e passar para o

estado $\mathfrak{f}(\mathfrak{faf}(\mathfrak{C}), @)$.

Este estado $\mathfrak{f}(\mathfrak{faf}(\mathfrak{C}), @)$ vem da m-função $\mathfrak{f}$ de dois argumentos, que aqui introduzimos, inspirada na m-função *find*. Descrição da atividade está na tabela.

Tabela 47 – Descrição de $\mathcal{M}$ a partir das imagens da m-função $\mathfrak{f}$

Estado		Símbolo	Operações	Estado final
$\mathfrak{f}(\mathfrak{C}, \alpha)$	{	ə	L	$\mathfrak{f}_1(\mathfrak{C}, \alpha)$
		Não ə	L	$\mathfrak{f}(\mathfrak{C}, \alpha)$
$\mathfrak{f}_1(\mathfrak{C}, \alpha)$	{	$\sqcup$		$\mathfrak{f}_2(\mathfrak{C}, \alpha)$
		Não $\sqcup$	L	$\mathfrak{f}(\mathfrak{C}, \alpha)$
$\mathfrak{f}_2(\mathfrak{C}, \alpha)$	{	α		$\mathfrak{C}$
		Não α	R	$\mathfrak{f}_2(\mathfrak{C}, \alpha)$

Fonte: Autor (2024)

Na nossa máquina, a partir deste estado, a cabeça procura o símbolo α mais à esquerda da fita. Também devemos falar da m-função $\mathfrak{faf}$, de *find another in front*. Descrição da atividade encontra-se na tabela.

Tabela 48 – Descrição de $\mathcal{M}$ a partir das imagens da m-função $\mathfrak{faf}$

Estado		Símbolo	Operações	Estado final
$\mathfrak{faf}(\mathfrak{C})$	{	@	R	$\mathfrak{faf}(\mathfrak{C})$
		β, não @ e não ə	P@, R	$\mathfrak{pnem}(\mathfrak{C}, \beta)$
		ə	P@, R	$\mathfrak{poe}(\mathfrak{C})$

Fonte: Autor (2024)

A partir desse estado, a máquina vai, começando em uma fila de símbolos @, encontrar o primeiro símbolo diferente. Se tal símbolo for ou não ə, então terá diferentes comportamentos, mas em ambos a máquina passa para um estado que leva tal símbolo para o fim da fita. Esta m-função $\mathfrak{pnem}$, de *print at the new end and mark*, tem a atividade respectiva descrita na tabela abaixo.

Este estado $\mathfrak{pnem}(\mathfrak{C}, \alpha)$ providencia que a máquina leve para o atual fim da fita, no lugar do símbolo %, o símbolo pós-@ escaneado durante o estado $\mathfrak{faf}(\mathfrak{C})$. Lá impresso, logo à direita dele a máquina imprime % novamente, como novo fim da fita, e muda para o estado $\mathfrak{f}(\mathfrak{faf}(\mathfrak{C}), @)$ novamente.

Tabela 49 – Descrição de $\mathcal{M}$ a partir das imagens da m-função $\mathfrak{pnem}$

Estado		Símbolo	Operações	Estado final
$\mathfrak{pnem}(\mathfrak{C}, \alpha)$	{	%	Pα, R, P%	$\mathfrak{f}(\mathfrak{faf}(\mathfrak{C}), @)$
		Não %	R	$\mathfrak{pnem}(\mathfrak{C}, \alpha)$

Fonte: Autor (2024)

E a m-função que denominamos $\mathfrak{poe}$, de *print old end*, causa a atividade descrita na tabela logo abaixo.

Tabela 50 – Descrição de $\mathcal{M}$ a partir da imagem da m-função $\mathfrak{poe}$

Estado		Símbolo	Operações	Estado final
$\mathfrak{poe}(\mathfrak{C})$	$\left\{\right.$	$\%$	P$\mathfrak{d}$, L	$\mathfrak{f}(\mathfrak{er}(\mathfrak{C}), @)$
		Não $\%$	R	$\mathfrak{poe}(\mathfrak{C})$

Fonte: Autor (2024)

Com esse estado, a máquina apagará o símbolo $\%$ ao fim da fita. O último símbolo será $\mathfrak{d}$ novamente. Nossa máquina de Turing, após isso, no estado $\mathfrak{f}(\mathfrak{er}(\mathfrak{C}), @)$, procura o primeiro símbolo $@$. E, com a cabeça em cima dele, passa para o estado $\mathfrak{er}(\mathfrak{C})$.

E definimos a m-função $\mathfrak{er}$, de *erase in a row*. A atividade respectiva está na tabela abaixo. A partir do estado $\mathfrak{er}(\mathfrak{C})$, nossa máquina de Turing apaga a fila de símbolos $@$ onde a cabeça se encontra desde o começo. E quando escaneia o símbolo pós-$@$, a máquina passa para o estado $\mathfrak{C}$.

Tabela 51 – Descrição de $\mathcal{M}$ a partir das imagens da m-função $\mathfrak{er}$

Estado		Símbolo	Operações	Estado final
$\mathfrak{er}(\mathfrak{C})$		Qualquer	P$\dot{\sqcup}$, R	$\mathfrak{er}_1(\mathfrak{C})$
$\mathfrak{er}_1(\mathfrak{C})$	$\left\{\right.$	$@$	P$\sqcup$, R	$\mathfrak{er}_1(\mathfrak{C})$
		Não $@$		$\mathfrak{C}$

Fonte: Autor (2024)

Parte D. Dito isso, após a máquina chegar à imagem $\left(\mathfrak{q}'_{3(k+1)}, \mathfrak{d}, \mathsf{L}\right)$ anteriormente indicada na parte **A)**, a cabeça de $\mathcal{M}$ vai retornar ao primeiro dos k símbolos sobreponteados, ou seja, a β_{-1}. Para isso, informalmente falando, faremos $\mathfrak{q}'_{3(k+1)} = \mathfrak{q}^+$.

Deste estado $\mathfrak{q}^+ \notin Q$, para o qual a máquina passa ao chegar no símbolo β_{-1}, definimos, também informalmente, a atividade provocada na máquina.

Tabela 52 – Descrição de $\mathcal{M}$ a partir do estado $\mathfrak{q}^+$

Estado		Símbolo	Operações	Estado final
$\mathfrak{q}^+$	$\left\{\right.$	$\mathfrak{d}$	R	$\mathfrak{q}_1^+$
		Não $\mathfrak{d}$	L	$\mathfrak{q}^+$
$\mathfrak{q}_1^+$	$\left\{\right.$	α_{m+i}		$\mathfrak{gw}(\mathfrak{q}_0, \alpha_i, \sqcup, \sqcup, \ldots, \sqcup)$
		Não α_{m+i}	R	$\mathfrak{q}_1^+$

Fonte: Autor (2024)

Aqui ao final, trata-se da m-função $\mathfrak{gw}$ de $(k+1)$ argumentos. E aí a máquina de uma só fita inicia a emulação das transições da máquina multifita, agindo em cada símbolo sobreponteado como $\mathcal{S}$ agiria em seu estado inicial. A máquina $\mathcal{M}$, como foi construída, chegará ao estado de aceitação exatamente nas mesmas entradas que levavam a outra máquina a ele.

O estado $\mathfrak{q}_{\text{aceita}}$ aparecerá aqui apenas quando toda a sequência de símbolos β_r e o argumento $\mathfrak{B}$ de $\mathfrak{gw}$ forem tais que a imagem de $\mathfrak{B}$ com os estados β_r na função δ nos levasse também a $\mathfrak{q}_{\text{aceita}}$. O mesmo vale para o estado $\mathfrak{q}_{\text{rejeita}}$. ■

Ainda é interessante notarmos que, embora haja tal equivalência, uma máquina multifita é mais rápida para realizar tarefas que uma máquina de Turing comum realizaria com sua única fita. Pela própria demonstração dada é visível que a máquina comum demora um tanto para replicar a ação da multifita.

4.2 MÁQUINA DE TURING NÃO-DETERMINÍSTICA

Outra variante importante é a máquina de Turing **não-determinística**. Ela é definida pela 7-upla $(Q, \Sigma, \Gamma, \delta, \mathfrak{q}_0, \mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}})$, com as mesmas condições que uma máquina de Turing comum, exceto que $\delta : Q \times \Gamma \to \mathcal{P}\left(Q \times \Gamma \times \{\mathsf{R}, \mathsf{L}\}\right)$.

De que se trata? A máquina lê um símbolo em algum estado e, em vez de se dirigir simplesmente para uma tríade de um símbolo, um estado e uma direção, ela dirigir-se-á para um conjunto de distintas tríades dessas. Isso significa que, após escanear uma configuração, a máquina se ramificará em diversas possibilidades. É como não ter apenas uma fita, mas sim fitas em paralelo, que surgem a partir dos escaneamentos anteriores e de quantas tríades aparecem após o escaneamento atual.

O funcionamento de uma máquina de Turing não-determinística pode ser considerado como uma árvore. E chamaremos de um **nó** dessa árvore cada configuração que surge de uma leitura da máquina. Assim, um nó é ou a configuração formada pelo estado $\mathfrak{q}_0$ junto do primeiro símbolo escaneado, ou é a configuração formada pelo estado de uma das tríades da imagem de uma configuração junto do símbolo escaneado após a cabeça se mover para a direção indicada na mesma tríade.

Dizemos que um segundo nó é **filho** de um primeiro outro nó se, e somente se ele é a configuração de uma das fitas paralelas advindas imediatamente do escaneamento do primeiro nó. Ao nó inicial de toda a nossa computação damos o nome de **raiz** da nossa "árvore". Consequentemente, o nó raiz não é filho de nenhum outro nó.

E dizemos que é um **ramo** dessa árvore cada sequência de nós, começando pela raiz, tal que, para $i \in \mathbb{N} - \{0\}$, o $(i+1)$-ésimo elemento dessa sequência é filho do i-ésimo elemento dela. Mas se deve observar que um ramo pode ser finito ou infinito. No caso de ser finito, haverá um último nó que não tem filhos. No caso de ser infinito, cada nó sempre tem um filho.

Diferente de uma máquina multifita, aqui vão surgindo novas configurações, junto de uma direção, cada uma com seu estado próprio, e independente das outras fitas em paralelo, a não ser pela questão do estado de aceitação. Isso porque, para

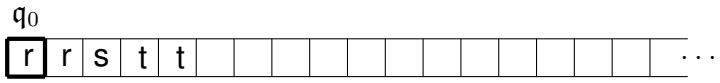
a máquina não-determinística, seu trabalho a partir de uma entrada termina se em qualquer um de seus nós houver o estado de aceitação.

Quando um nó tem o estado de rejeição, ele dá fim ao seu ramo, mas não aos outros ramos da máquina. O mesmo acontece quando a imagem de um nó, pela função de transição, é o conjunto vazio.

Exemplo 8. Vamos ver como funciona a máquina $\mathcal{M}$ cujo alfabeto da fita é $\Gamma = \{\sqcup, \mathsf{r}, \mathsf{s}, \mathsf{t}\}$, cujo alfabeto de entrada é $\Sigma = \Gamma - \{\sqcup\}$, cujo conjunto de estados é $Q = \{\mathfrak{q}_0, \mathfrak{q}_1, \mathfrak{q}_2, \mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}}\}$, o estado inicial é $\mathfrak{q}_0$ e que possui uma função de transição $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{\mathsf{R}, \mathsf{L}\})$.

Para visualizarmos, tomaremos como entrada, a cadeia rrstt. A máquina inicia escaneando o primeiro quadrado, com o estado $\mathfrak{q}_0$. Para ajudar o acompanhamento de cada nó, colocaremos o estado atual acima do quadrado escaneado em cada figura.

Figura 24 – A raiz no exemplo de máquina de Turing não-determinística



Fonte: Autor (2024)

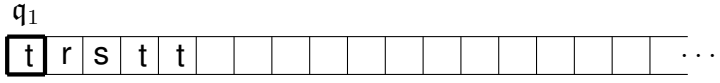
Suponhamos, neste exemplo, que nossa função de transição é tal que $\delta(\mathfrak{q}_0, \mathsf{r}) = \{(\mathfrak{q}_1, \sqcup, \mathsf{R}), (\mathfrak{q}_1, \mathsf{t}, \mathsf{L}), (\mathfrak{q}_0, \mathsf{s}, \mathsf{R})\}$. Isso vai dar origem a três nós filhos da raiz. Para o primeiro destes filhos, teremos a fita paralela abaixo representada, que se encontra no estado $\mathfrak{q}_1$.

Figura 25 – Primeiro filho da raiz no exemplo

Fonte: Autor (2024)

Para o segundo filho da raiz, teremos a fita paralela abaixo representada, a qual também se encontra no estado $\mathfrak{q}_1$. O quadrado escaneado pela cabeça ainda é o mesmo devido ao fato de que o último deslocamento se deu do primeiro quadrado da fita para o sentido da esquerda.

Figura 26 – Segundo filho da raiz no exemplo

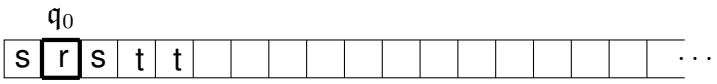


Fonte: Autor (2024)

E em seguida, representamos a fita paralela para o terceiro e último filho da raiz. Esta, diferente das fitas anteriores, se encontra no estado $\mathfrak{q}_0$. Com o que vemos,

não havemos de nos espantar se cada uma dessas fitas der origem a caminhos muito distintos uns dos outros nas computações que ainda virão.

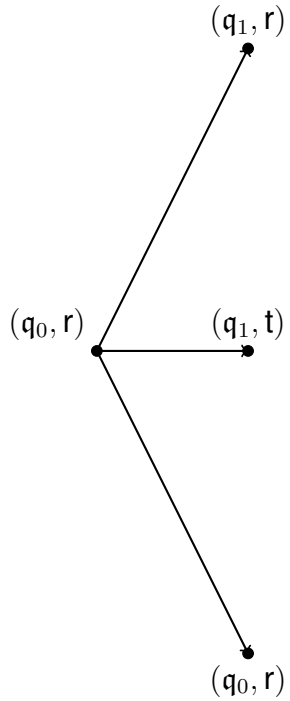
Figura 27 – Terceiro filho da raiz no exemplo



Fonte: Autor (2024)

Vemos que os três nós que surgiram equivalem às configurações (q_1, r), (q_1, t) e (q_0, r). Podemos fazer um diagrama de árvore dos nós percorridos pela computação de nossa máquina de Turing até o momento. Segue na figura abaixo, nosso diagrama para tal, desenhado no sentido da esquerda para a direita.

Figura 28 – Árvore da computação no exemplo (parte 1)



Fonte: Autor (2024)

Dados os três filhos tidos a partir da raiz, estes três darão seguimento à computação em questão. E tal computação seguirá funcionando pelo mesmo princípio.

Assim sendo, da transição com base na nossa raiz surgiram três fitas consideradas, a princípio, separadamente. E cada uma das fitas paralelas será submetida à ação da máquina não-determinística.

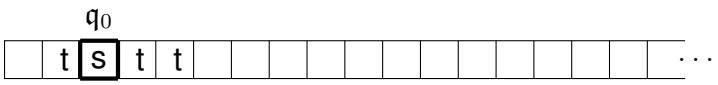
Não dizemos que tais fitas são consideradas de modo separado em absoluto apenas pela questão do estado de aceitação. Afinal, se uma delas chegar a este estado e as outras não, ainda assim as outras finalizam seu trabalho em função desta uma. Se não for isso, cada uma delas segue funcionando de forma independente.

Já notamos uma grande diferença entre a máquina de Turing não-determinística e a máquina de Turing (determinística) multifita. Em ambas podemos falar, em certo sentido, de várias fitas guardando cada uma seus símbolos e transições entre símbolos com base em estados.

Contudo, na máquina multifita, o símbolo escaneado em uma dessas fitas interfere em qual será o estado sucessor a partir do qual a máquina operará em todas as fitas presentes. Enquanto que, na máquina não-determinística, cada fita seguirá o próprio caminho, a partir de seus símbolos e do estado específico em que se encontram.

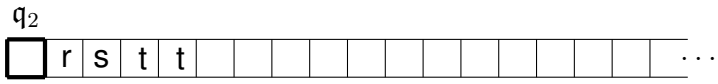
Dada a fita do primeiro nó dentre os três filhos, suponhamos que $\delta(\mathfrak{q}_1, \mathsf{r}) = \{(\mathfrak{q}_0, \mathsf{t}, \mathsf{R}), (\mathfrak{q}_2, \mathsf{r}, \mathsf{L})\}$. Então, teremos a origem de dois nós, e com eles, duas fitas paralelas. Abaixo, a primeira delas, que se encontra no estado $\mathfrak{q}_0$. E na sequência, a segunda delas, que se encontra no estado $\mathfrak{q}_2$.

Figura 29 – Primeiro “neto” da raiz no exemplo



Fonte: Autor (2024)

Figura 30 – Segundo “neto” da raiz no exemplo

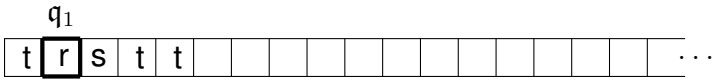


Fonte: Autor (2024)

Depois disso, nos dirigimos para o segundo dentre os nossos três nós anteriores. Suponhamos, então, que $\delta(\mathfrak{q}_1, \mathsf{t}) = \{(\mathfrak{q}_1, \mathsf{t}, \mathsf{R}), (\mathfrak{q}_1, \mathsf{t}, \mathsf{L}), (\mathfrak{q}_2, \textvisiblespace, \mathsf{R}), (\mathfrak{q}_{\text{rejeita}}, \mathsf{t}, \mathsf{R})\}$. Aqui teremos a origem de mais quatro nós, filhos do segundo filho da raiz. Deste modo, presenciamos como a computação continua, a partir de mais quatro fitas paralelas.

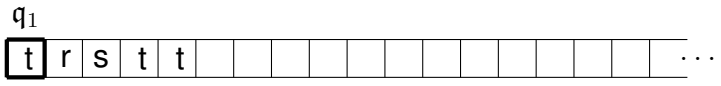
Abaixo, nós vemos a primeira dessas fitas paralelas, a qual se encontra no estado $\mathfrak{q}_1$. A segunda dessas fitas também se encontra no estado $\mathfrak{q}_2$. Já a terceira fita paralela se encontra no estado $\mathfrak{q}_3$. Enquanto que a quarta fita se encontra no estado $\mathfrak{q}_{\text{rejeita}}$.

Figura 31 – Terceiro “neto” da raiz no exemplo



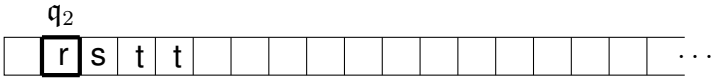
Fonte: Autor (2024)

Figura 32 – Quarto “neto” da raiz no exemplo



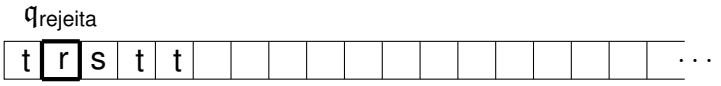
Fonte: Autor (2024)

Figura 33 – Quinto “neto” da raiz no exemplo



Fonte: Autor (2024)

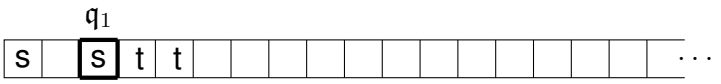
Figura 34 – Sexto “neto” da raiz no exemplo



Fonte: Autor (2024)

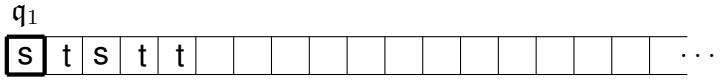
E do terceiro de nossos três nós antigos, já conhecemos a transição. É a mesma configuração da raiz. Mais três nós, três fitas paralelas. A primeira, no estado q_1. A segunda, também no estado q_1. E a terceira, no estado q_2.

Figura 35 – Sétimo “neto” da raiz no exemplo



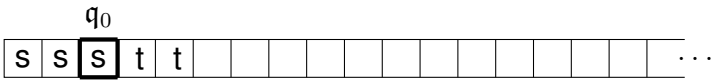
Fonte: Autor (2024)

Figura 36 – Oitavo “neto” da raiz no exemplo



Fonte: Autor (2024)

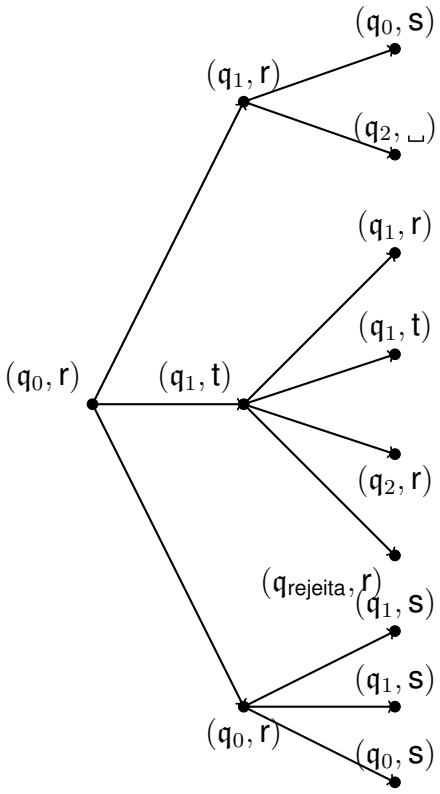
Figura 37 – Nono “neto” da raiz no exemplo



Fonte: Autor (2024)

E a partir das transições agora contempladas na máquina, o diagrama da árvore ficaria como a seguir.

Figura 38 – Árvore da computação no exemplo (parte 2)



Fonte: Autor (2024)

Na próxima etapa, os novos nós terão seus filhos, exceto pelo nó $(q_{rejeita}, r)$. Ali marca o fim de um ramo. E já conhecemos a transição das configurações (q_1, t) e (q_1, r).

Supondo que $\delta(q_0, s) = \{(q_0, s, R), (q_0, \sqcup, L), (q_1, t, L), (q_2, s, R), (q_{rejeita}, s, R)\}$, e também que $\delta(q_2, \sqcup) = \varnothing$, $\delta(q_2, r) = \{(q_2, \sqcup, R), (q_2, s, R)\}$, e $\delta(q_1, s) = \{(q_0, t, R)\}$, poderemos ver como o trabalho da máquina continua. Mas o façamos sem recorrer à visão de cada uma das fitas paralelas que surgem.

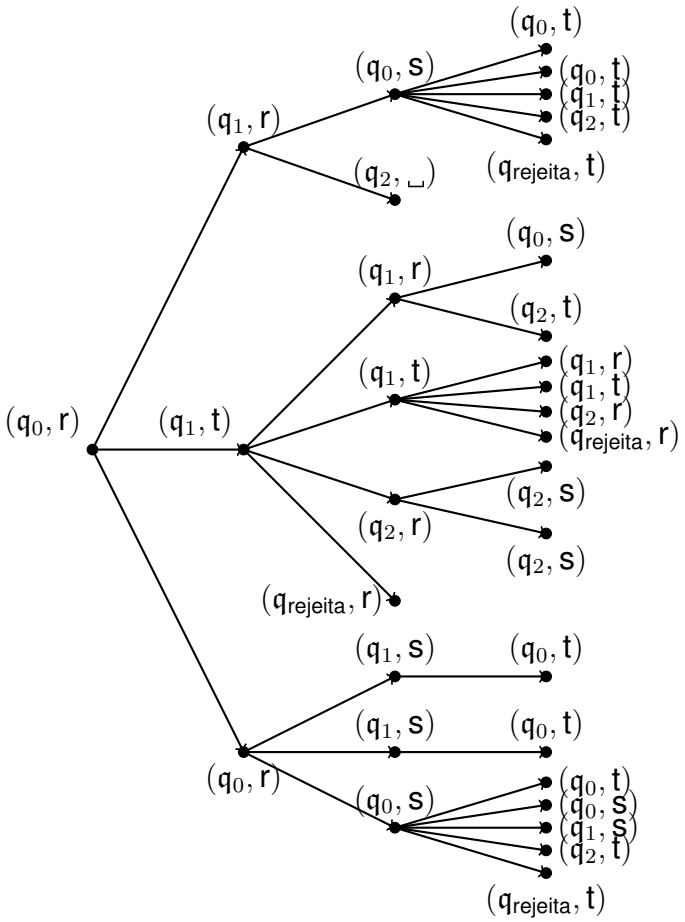
A partir dos nove nós que surgiram por último, em vez de explicitarmos todas as respectivas configurações completas, seguiremos desenhando a árvore que representa a computação de nossa máquina não-determinística. Surgem, neste momento, mais vinte filhos ao total.

O diagrama da árvore para a computação, com as transições contempladas agora, ficaria como na última figura deste exemplo, abaixo.

E assim a máquina segue. Ela continuará trabalhando até que algum dos nós contenha o estado de aceitação. Se isso acontecer, todos os ramos findarão naquela etapa. E se isso não acontecer, eles prosseguirão indefinidamente. Como vimos, alguns ramos são interrompidos por si mesmos.

E, como podemos notar, se o estado de aceitação não se fizer presente a partir dessa entrada, haverá pelo menos um ramo que se estenderá ao infinito. É o caso,

Figura 39 – Árvore da computação no exemplo (parte 3)



Fonte: Autor (2024)

aqui, do ramo de seguidos nós (q_1, t) logo após a raiz. Pois, analisando tal fita particular, vemos que a cabeça da máquina está presa sempre no primeiro quadrado, imprimindo o símbolo t, passando sempre para o estado q_1. E dessa situação não sairá se a máquina não parar sua atividade por completo. □

Cabe observarmos que as máquinas de Turing comuns também podem ser consideradas não-determinísticas. Pois elas são tais que, as imagens da função de transição, embora elementos daquele conjunto das partes, podem ser todas conjuntos de um só elemento.

Ou seja, neste caso, para todos $q \in Q$ e $\alpha \in \Gamma$, haverá algum $q' \in Q$, algum $\alpha' \in \Gamma$ e algum $X \in \{R, L\}$ com os quais sempre teremos $\delta(q, \alpha) = \{(q', \alpha', X)\}$. Isto significa que a máquina terá um só ramo. Ela não terá múltiplas possibilidades de fita, mas uma fita só.

Chamando as máquinas de Turing comuns de máquinas determinísticas, podemos afirmar o seguinte resultado.

Teorema 4.2.1. *Toda máquina de Turing não-determinística é equivalente a alguma*

máquina de Turing determinística.

Demonstração. A estratégia utilizada aqui para reproduzir o trabalho de uma máquina de Turing não-determinística é fazer uma "busca em largura" através dos seus ramos, em vez de uma "busca em profundidade". A busca em profundidade consistiria em explorar cada ramo em sua totalidade, desde a raiz até o seu fim. Só que ramos podem ser infinitos, e isso facilmente impediria de encontrarmos o estado de aceitação em outro ramo enquanto estivermos em um ramo infinito.

Já a busca em largura consistirá em explorarmos os ramos sempre até os nós que estão "a uma mesma distância" da raiz. Queremos dizer, com isso, explorar primeiro os ramos indo até os nós filhos da raiz (que são em quantidade finita); depois os ramos indo até os filhos dos filhos da raiz (também em quantidade finita); depois os ramos indo até os filhos dos filhos dos filhos da raiz (quantidade finita); e assim por diante. Deste modo, exploraremos todos os nós da máquina não-determinística. Se em algum deles encontrarmos o estado de aceitação, aceitaremos a entrada.

Seja, $\mathcal{N} = (Q, \Sigma, \Gamma, \delta, \mathfrak{q}_0, \mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}})$ uma máquina de Turing não-determinística qualquer. Temos o alfabeto da fita $\Gamma = \{\alpha_0, \alpha_1, \ldots, \alpha_{m-1}\}$ e o conjunto de estados $Q = \{\mathfrak{q}_0, \mathfrak{q}_1, \ldots, \mathfrak{q}_{n-1}\}$.

Primeiramente, vamos aos casos mais simples. Se $\mathfrak{q}_0 = \mathfrak{q}_{\text{aceita}}$, então basta tomarmos uma máquina de Turing determinística com o mesmo alfabeto de entrada e cujo estado inicial também seja o estado de aceitação. Se $\mathfrak{q}_0 = \mathfrak{q}_{\text{rejeita}}$, o mesmo, mas com o estado de rejeição.

Agora, nos outros casos, vamos nos valer do teorema 4.1.1, o qual nos afirma a equivalência entre uma máquina multifita e alguma máquina de uma única fita. Construiremos uma máquina de três fitas equivalente à nossa máquina não-determinística.

Sejam $b \in \mathbb{N}$ o número de elementos do conjunto $Q \times \Gamma \times \{\mathsf{R}, \mathsf{L}\}$ e $@$ um símbolo que não pertença a Γ. Agora, tomemos cada símbolo de Γ e, a partir de cada um, criaremos um novo símbolo com um ponto escrito acima do símbolo original. Assim, a partir de α_0, criamos o símbolo $\dot{\alpha}_0$. A partir de α_1, criamos o símbolo $\dot{\alpha}_1$. E assim seguimos até o símbolo α_{m-1}, a partir do qual criamos o símbolo $\dot{\alpha}_{m-1}$.

Seja, então, $\Gamma' = \Gamma \cup \{\dot{\alpha}_0, \dot{\alpha}_1, \ldots, \dot{\alpha}_{m-1}, @, 1, 2, 3, \ldots, b\}$. A máquina equivalente a $\mathcal{N}$ que apresentaremos é

$$\mathcal{S} = \left(Q', \Sigma, \Gamma', \delta', \mathfrak{q}'_0, \mathfrak{q}'_{\text{aceita}}, \mathfrak{q}'_{\text{rejeita}}\right),$$

onde $\delta' : Q' \times (\Gamma')^3 \to Q \times (\Gamma')^3 \times \{\mathsf{R}, \mathsf{L}\}^3$. Seus estados e sua função de transição serão especificados abaixo.

Vamos organizar as tríades do conjunto $Q \times \Gamma \times \{\mathsf{R}, \mathsf{L}\}$ enumerando-as de 1 até b. A nossa máquina de três fitas será tal que sua primeira fita receberá a entrada, e seus quadrados nunca receberão mudança. A terceira fita vai registrar qual "ramo"

da máquina $\mathcal{N}$ estamos explorando. Seu primeiro quadrado ficará vazio, para marcar o início da fita. Mas será seguido por uma cadeia de símbolos da linguagem $B = \{1, 2, 3, \ldots, b\}$.

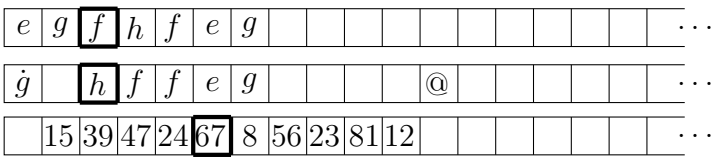
Esta cadeia mudará seguidamente, passando primeiremente pelas possibilidades com apenas um símbolo, depois pelas possibilidades com dois símbolos, depois com três, depois com quatro, e assim por diante. E, dentro de um mesmo comprimento, as cadeias mudarão, a princípio, segundo a ordem lexicográfica. Tal cadeia representa por quais possibilidades de tríades estado/símbolo/direção passamos desde a raiz. Definimos que a tríade número 1 será $(\mathfrak{q}_0, \alpha_0, \mathsf{R})$.

Por exemplo, a cadeia $2\,4\,6\,5$ na terceira fita vai indicar o "ramo" que, após a raiz, passa para a segunda tríade de $Q \times \Gamma \times \{\mathsf{R}, \mathsf{L}\}$, da segunda passa para a quarta, da quarta para a sexta e da sexta para a quinta. Falamos de ramo entre aspas pois nem todas estas tríades estão presentes, de fato, naquele lugar da computação de $\mathcal{N}$. Apenas estamos passando por esse ramo em potencial, mas o que vai distinguir se tudo nele faz parte do trabalho de $\mathcal{N}$ serão, como veremos, os estados da máquina $\mathcal{S}$.

A segunda fita irá simular o trabalho que cada ramo de $\mathcal{N}$ teria com a cadeia de entrada. Ele contará com o símbolo $@$ à sua extrema direita para marcar o fim da fita. E o primeiro símbolo da segunda fita ficará sempre sobreponteado, enquanto os outros não, para marcar o início da fita.

Por exemplo, abaixo a representação de um estágio de uma máquina $\mathcal{S}$, que, equivalente a uma máquina $\mathcal{N}$, recebe a entrada $egfhfeg$.

Figura 40 – Máquina de 3 fitas equivalente a uma máquina não-determinística



Fonte: Autor (2024)

O que a máquina está explorando agora é o ramo por onde a máquina $\mathcal{N}$ teria passado, respectivamente, pela tríade de número 15, pela de número 39, pela 47, pela 24, pela 67, e que, no momento, ainda deve se estender passando pela 8, pela 56, pela 23, pela 81 e pela 12. A simulação na segunda fita da máquina já andou 4 unidades e está prestes a andar pela quinta vez.

Aqui vale observar que, embora haja mais de um dígito em vários dos quadrados da terceira fita, na verdade se trata de um símbolo só. O conteúdo 15 no segundo quadrado não são um símbolo 1 e um símbolo 5, mas um único símbolo 15. O mesmo para os semelhantes.

Atualmente estamos vendo os ramos em até 10 nós "de distância da raiz". E se estamos nesta numeração vista na terceira fita, significa que ainda não passamos por

um ramo com estado de aceitação que estivesse presente na máquina $\mathcal{N}$.

Mostrado isto, vamos às especificações. Em todas as transições, a direção do movimento na primeira fita será igual à direção na segunda fita. Isto se dá para que as duas estejam sempre juntas a fim de que a segunda fita sempre possa receber novamente uma cópia da cadeia de entrada.

Os primeiros estados da máquina $\mathcal{S}$ servem para copiar a entrada da primeira fita para a segunda, marcar o fim desta com $@$, e imprimir o 1 para começar a cadeia no segundo quadrado da terceira fita.

$\delta'(\mathfrak{q}'_0, \beta_1, \beta_2, \beta_3) = (\mathfrak{q}'_1, \beta_1, \beta_2, \beta_3, \mathsf{R}, \mathsf{R}, \mathsf{L})$ para $\beta_1 \neq \sqcup$.
$\delta'(\mathfrak{q}'_0, \sqcup, \beta_2, \beta_3) = (\mathfrak{q}'_4, \sqcup, \beta_2, \beta_3, \mathsf{R}, \mathsf{R}, \mathsf{L})$.
$\delta'(\mathfrak{q}'_1, \beta_1, \beta_2, \beta_3) = (\mathfrak{q}'_1, \beta_1, \beta_2, \beta_3, \mathsf{R}, \mathsf{R}, \mathsf{L})$ para $\beta_1 \neq \sqcup$.
$\delta'(\mathfrak{q}'_1, \sqcup, \beta_2, \beta_3) = (\mathfrak{q}'_2, \sqcup, \beta_2, \beta_3, \mathsf{R}, \mathsf{R}, \mathsf{L})$.
$\delta'(\mathfrak{q}'_2, \beta_1, \beta_2, \beta_3) = (\mathfrak{q}'_3, \beta_1, @, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{L})$.
$\delta'(\mathfrak{q}'_3, \beta_1, \sqcup, \beta_3) = (\mathfrak{q}'_3, \beta_1, \beta_1, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{L})$.
$\delta'(\mathfrak{q}'_3, \beta_1, \beta_2, \beta_3) = \left(\mathfrak{q}'_8, \beta_1, \dot{\beta}_2, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$ para $\beta_2 \in \Sigma$.
$\delta'(\mathfrak{q}'_4, \beta_1, \beta_2, \beta_3) = (\mathfrak{q}'_5, \beta_1, \beta_2, \beta_3, \mathsf{R}, \mathsf{R}, \mathsf{L})$.
$\delta'(\mathfrak{q}'_5, \beta_1, \beta_2, \beta_3) = (\mathfrak{q}'_6, \beta_1, @, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{L})$.
$\delta'(\mathfrak{q}'_6, \beta_1, \beta_2, \beta_3) = (\mathfrak{q}'_7, \beta_1, \beta_2, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{L})$.
$\delta'(\mathfrak{q}'_7, \beta_1, \beta_2, \beta_3) = (\mathfrak{q}'_8, \beta_1, \dot{\sqcup}, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{R})$.
$\delta'\left(\mathfrak{q}_8, \beta_1, \dot{\beta}_2, \beta_3\right) = \left(\mathfrak{q}^{1,0}_{0,\beta_2}, \beta_1, \dot{\beta}_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$ para $\beta_2 \in \Gamma$.

Agora, para todos os estados que forem descritos a seguir, fica definido o seguinte. Para todo estado $\mathfrak{q} \neq \mathfrak{q}'_9$ especificado abaixo, ficam definidos outros três estados $(\mathfrak{q})_1$, $(\mathfrak{q})_2$ e $(\mathfrak{q})_3$, para os quais:

$\delta'(\mathfrak{q}, \beta_1, @, \beta_3) = ((\mathfrak{q})_1, \beta_1, \sqcup, \beta_3, \mathsf{R}, \mathsf{R}, \mathsf{R})$.
$\delta'((\mathfrak{q})_1, \beta_1, \beta_2, \beta_3) = ((\mathfrak{q})_2, \beta_1, \sqcup, \beta_3, \mathsf{R}, \mathsf{R}, \mathsf{R})$.
$\delta'((\mathfrak{q})_2, \beta_1, \beta_2, \beta_3) = ((\mathfrak{q})_3, \beta_1, @, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{L})$.
$\delta'((\mathfrak{q})_3, \beta_1, \beta_2, \beta_3) = (\mathfrak{q}, \beta_1, \beta_2, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{L})$.

Isto serve para que, se em algum momento durante a simulação na segunda fita a cabeça esbarrar com $@$, a máquina leva o $@$ duas unidades para a direita e depois continua sua simulação normalmente. Deixaremos essa informação implícita daqui para frente, então não mencionaremos o caso em que a máquina, nos estados seguintes, lerá o símbolo $@$.

Ao final da transição a partir do estado $\mathfrak{q}'_8$, colocamos que o estado sucessor era $\mathfrak{q}^{1,0}_{0,\beta_2}$. É a partir daqui que se encontra o coração da nossa máquina $\mathcal{S}$. Explicaremos sua notação mais à frente.

Também devemos mencionar os estados $\mathfrak{q}'_9$ ao $\mathfrak{q}'_{12}$. Eles serão os responsáveis por fazer $\mathcal{S}$ recomeçar seu trabalho, passando para o próximo ramo depois de ter explorado um ramo anterior. No uso desses estados, a máquina encontra o fim da segunda fita, marcado por $@$, o qual sempre estará à direita do fim da cadeia de

entrada na primeira fita, para então reimprimir a cadeia de entrada da primeira fita na segunda. Nisso, voltamos ao começo da segunda (e da primeira) fita. E, por fim, encontram o início da cadeia na terceira fita para que a máquina continue seu trabalho.

$\delta'(\mathfrak{q}'_9, \beta_1, \beta_2, \beta_3) = (\mathfrak{q}'_9, \beta_1, \beta_2, , \beta_3, \mathsf{R}, \mathsf{R}, \mathsf{L})$ para $\beta_2 \neq @$.
$\delta'(\mathfrak{q}'_9, \beta_1, @, \beta_3) = (\mathfrak{q}'_{10}, \beta_1, @, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{L})$.
$\delta'(\mathfrak{q}'_{10}, \beta_1, \beta_2, \beta_3) = (\mathfrak{q}'_{10}, \beta_1, \beta_1, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{L})$ para $\beta_2 \in \Gamma$.
$\delta'\left(\mathfrak{q}'_{10}, \beta_1, \dot{\beta}_2, \beta_3\right) = \left(\mathfrak{q}'_{11}, \beta_1, \dot{\beta}_1, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{L}\right)$ para $\beta_2 \in \Gamma$.
$\delta'(\mathfrak{q}'_{11}, \beta_1, \beta_2, \beta_3) = (\mathfrak{q}'_{11}, \beta_1, \beta_2, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{L})$ para $\beta_3 \neq \sqcup$.
$\delta'(\mathfrak{q}'_{11}, \beta_1, \beta_2, \sqcup) = (\mathfrak{q}'_{11}, \beta_1, \beta_2, \sqcup, \mathsf{L}, \mathsf{L}, \mathsf{R})$.
$\delta'\left(\mathfrak{q}'_{12}, \beta_1, \dot{\beta}_2, r\right) = \left(\mathfrak{q}^{r,0}_{0,\beta_2}, \beta_1, \dot{\beta}_2, r, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$.

Agora explicaremos a notação antes mencionada. Sejam $\beta \in \Gamma$ e $i, k \in \mathbb{N}$, com $0 \leq i \leq n-1$ e $1 \leq k \leq b$. Chamemos de t_k a k-ésima tríade do conjunto $Q \times \Gamma \times \{\mathsf{R}, \mathsf{L}\}$ tal qual o organizamos. Então, temos que, para algum k acontece que $\delta(\mathfrak{q}_i, \beta) = \{t_{l_1}, t_{l_2}, \ldots, t_{l_k}\}$, onde l_1, l_2, ..., l_k são números distintos de 1 a b, ou $\delta(\mathfrak{q}_i, \beta) = \varnothing$. A partir disso, definimos o conjunto

$$B_{i,\beta} = \begin{cases} \{l_1, l_2, \ldots, l_k\}, & \text{se } \delta(\mathfrak{q}_i, \beta) \neq \varnothing, \\ \varnothing, & \text{se } \delta(\mathfrak{q}_i, \beta) = \varnothing \end{cases}.$$

Agora, seja $r \in B$. Um estado $\mathfrak{q}^{r,0}_{i,\beta}$ indica que vamos explorar a r-ésima tríade, sendo que r pode pertencer ao conjunto $B_{i,\beta}$ ou não. Se pertencer, o estado terá uma determinada postura nas transições. Se não pertencer, ele terá outra postura. O número 0 ao lado de r está ali porque este é o primeiro estado que traz essas informações relacionando a r-ésima tríade com a configuração $(\mathfrak{q}_i, \beta)$. Haverá outros em seguida.

O r de tais estados é definido pelo número da cadeia na terceira fita a partir do qual a máquina $\mathcal{S}$ vai simular o trabalho de $\mathcal{N}$ no momento. Como já se sabe que a primeira cadeia de todas a ser explorada é a 1, e o será no primeiro quadrado da segunda fita, do estado $\mathfrak{q}'_8$ já se partiu para o estado $\mathfrak{q}^{1,0}_{0,\beta_2}$. Tal estado indica que vamos explorar a primeira tríade, e a necessidade de tal simulação dependerá da condição de ela estar presente no conjunto do $\delta(\mathfrak{q}_0, \beta_2)$, que era a configuração da raiz.

Para um $r \in B$, seja a função $f : B \to \{0, 1, \ldots, n-1\}$ tal que leva r até o número do estado de sua tríade. Com isso, o estado da r-ésima tríade se trata do estado $\mathfrak{q}_{f(r)}$.

Sendo assim, dividimos em oito casos as possibilidades de estados $\mathfrak{q}^{r,0}_{i,\beta}$ com suas transições. A seguir apresentados os casos, deixamos as especificações de suas transições para o apêndice A.

1. Comecemos pelo caso em que $r \in B_{i,\beta}$, com $r \neq b$, e também $\mathfrak{q}_{f(r)} \neq \mathfrak{q}_{\text{aceita}}$ e $\mathfrak{q}_{f(r)} \neq \mathfrak{q}_{\text{rejeita}}$.

A partir de tais estados a máquina simula, na segunda fita, a atividade de $\mathcal{N}$ no ramo indicado. Imprime os símbolos correspondentes aos que $\mathcal{N}$ imprimiria. E se move nas direções em que $\mathcal{N}$ se moveria. Em seguida, na terceira fita, anda para a direita a fim de continuar a explorar o ramo.

Explorá-lo-á, se ele continuar, em vista do símbolo escaneado anteriormente na segunda fita e do estado de $\mathcal{N}$ correlacionado aos estados em que antes a máquina $\mathcal{S}$ se encontrava. Pois ambos correspondem ao nó de $\mathcal{N}$, a partir do qual saberemos quais seriam os estados sucessores na transição da máquina não-determinística.

Se tal r é o último símbolo da cadeia da terceira fita, chegamos ao final do ramo presente. Tal cadeia, então, não termina com b, e nestes estados a máquina troca o símbolo ao final pelo seu sucessor. E então passa para o estado $\mathfrak{q}'_9$.

2. Vejamos o caso em que $r \in B_{i,\beta}$, com $r = b$, e também $\mathfrak{q}_{f(r)} \neq \mathfrak{q}_{\text{aceita}}$ e $\mathfrak{q}_{f(r)} \neq \mathfrak{q}_{\text{rejeita}}$.

   A máquina ainda simula $\mathcal{N}$ na segunda fita. Mas logo depois verifica se o b da terceira fita está no final da cadeia ou não.

   Se não está no final, então a máquina segue como faria no caso anterior. Se está no final, então, na terceira fita, ela volta até o último símbolo diferente de b para trocá-lo por seu sucessor, enquanto troca os símbolos b por 1. Mas se toda a cadeia é formada por símbolos b, então ela troca todos por 1 e adiciona mais um 1 à direita no fim da fita.

   Nessa situação, aumentamos o comprimento da cadeia da terceira da fita em uma unidade para começar a explorar os ramos com mais um filho, segundo a ordem lexicográfica. Feito isso, a máquina passa para o estado $\mathfrak{q}'_9$.

3. Dado isso, agora vamos ao caso em que $r \in B_{i,\beta}$, com $\mathfrak{q}_{f(r)} = \mathfrak{q}_{\text{rejeita}}$ e $r \neq b$.

   Aqui, como o ramo se encontra em um estado de rejeição que estaria presente na atividade de $\mathcal{N}$, cancelamos a simulação dele. E vamos além, pois cancelamos também todos os ramos naquele tamanho que passavam por aquele nó com estado de rejeição. Pulamos vários elementos na ordem lexicográfica, os quais seriam rejeitados na máquina não-determinística, e partimos já para um ramo no mesmo tamanho, mas sem este nó.

   Para isto, além de trocarmos o símbolo da nossa tríade na cadeia da terceira fita por seu sucessor, trocamos todos os números à sua direita por 1. E passamos para o estado $\mathfrak{q}'_9$.

4. Vamos, então, para o caso em que $r \in B_{i,\beta}$, com $\mathfrak{q}_{f(r)} = \mathfrak{q}_{\text{rejeita}}$ e $r = b$. Aqui a máquina vai ter novamente o aproveitamento de pular elementos da ordem

lexicográfica.

Porém, como tal nó com estado de rejeição se encontra na b-ésima tríade, então a cadeia da terceira fita terá ainda que ser modificada em certos símbolos anteriores a tal b, como também trocar todos os números à frente de tal b por 1.

Além de que há a possibilidade de, até aquele nó, a cadeia ser formada toda por símbolos b. Isso será verificado e, tal como no caso 2, se assim for, além de substituirmos todos os números por 1, ainda acrescentaremos outro 1 no fim da fita.

E claro, ao final do procedimento com tais estados, a máquina passa para o estado $\mathfrak{q}'_9$.

5. Há o caso em que, para $B_{i,\beta} \neq \varnothing$, temos $r \notin B_{i,\beta}$, com $r \neq b$. Mas na verdade temos aqui a mesma lógica do caso 3.

   Afinal, uma tríade que não está em um ramo válido vai cancelar a simulação de tal ramo na máquina multifita, assim como o faria a tríade cujo estado fosse de rejeição. E sabemos que os outros ramos que tivessem tal nó naquela posição também seriam inválidos.

   Podemos seguir, daí, o mesmo procedimento que o caso 3 faria, inclusive com relação à sua cadeia da terceira fita e o sucessor do símbolo r. E passamos para o estado $\mathfrak{q}'_9$.

6. Há também o caso em que, para $B_{i,\beta} \neq \varnothing$, temos $r \notin B_{i,\beta}$, com $r = b$. Só que temos aqui a mesma lógica do caso 4.

   A explicação é a mesma que foi feita no caso 6, com a diferença de que ocorre a devida verificação a partir do símbolo b. E faremos tal procedimento ou para apenas darmos o pulo na ordem lexicográfica, ou para já iniciarmos a explorar os ramos com maior comprimento. E passamos para $\mathfrak{q}'_9$.

7. Temos o caso em que $B_{i,\beta} = \varnothing$. Neste caso basta definirmos os estados $\mathfrak{q}_{i,\beta}^{r,0}$ e semelhantes para $r = 1$. Pois é a partir da primeira tríade que exploraremos tal caso e, como ele é tal que não possui tríade nenhuma, mal entramos e já saímos dele.

   Assim como nos casos 3 e 4, aqui temos um pulo na ordem lexicográfica. Esse pulo exclui todos os ramos daquele tamanho que possuam aquele nó cuja imagem é vazia na função de transição de $\mathcal{N}$. E, feito isso, passamos para $\mathfrak{q}'_9$.

8. Por fim, o caso em que $r \in B_{i,\beta}$, com $\mathfrak{q}_{f(r)} = \mathfrak{q}_{\text{aceita}}$. Quando isso acontece, devemos ir para o estado de aceitação de $\mathcal{S}$. Dado que é simples, abaixo, especificamos sua transição.

$$\delta'\left(\mathfrak{q}_{i,\beta}^{r,0}, \beta_1, \beta_2, \beta_3\right) = (\mathfrak{q}'_{\text{aceita}}, \beta_1, \beta_2, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{L}).$$

Feitas tais descrições de todos os casos, aquilo que não foi especificado dos estados acima e suas transições fica livre para ser definido, visto que não afetará no andamento da máquina. Assim, nossa máquina $\mathcal{S}$ vai simular os estados pelos quais $\mathcal{N}$ passaria com os respectivos símbolos na segunda fita. Vai ignorar os nós por que $\mathcal{N}$ não passaria.

Quando $\mathcal{N}$ aceitasse uma entrada, $\mathcal{S}$ também vai aceitar, visto que vai simular o trabalho de $\mathcal{N}$ até um nó dela que tenha o estado de aceitação e, consequentemente, aceitar. E quando $\mathcal{N}$ não aceitasse uma entrada, não chegando nunca ao estado de aceitação, o mesmo vai acontecer com $\mathcal{S}$, que só atinge a sua aceitação quando um nó de $\mathcal{N}$ atinge a sua.

Portanto, a máquina $\mathcal{S}$ é equivalente à máquina $\mathcal{N}$. ■

4.3 A MÁQUINA UNIVERSAL

Retornando nossos olhares para a classe das máquinas $\mathcal{M}^{\mathcal{T}}$, há uma máquina em particular que merece nossa atenção. Em verdade, a máquina de que vamos falar não se encontra dentro de tal classe, do modo como a definimos, simplesmente porque o alfabeto de entrada desta máquina não é o conjunto vazio.

Trata-se da chamada **máquina de Turing universal**, ou simplesmente máquina universal. Denotada aqui por $\mathcal{U}^{\mathcal{T}}$, ela é a máquina capaz de replicar, ao longo da sua fita, a computação dos F-símbolos que uma máquina $\mathcal{M}^{\mathcal{T}}$ qualquer faria.

O conceito da máquina universal foi muito importante no desenvolvimento dos primeiros computadores. A "existência" de uma máquina como $\mathcal{U}^{\mathcal{T}}$ não é óbvia. Turing teve de mostrar como ela funcionaria. E, fisicamente falando, ela é importantíssima dentro da área da computação, devido ao fato de que consegue simular outras máquinas.

Ela trabalha a partir da descrição padrão de uma máquina para replicar sua sequência computável. Portanto, as entradas da máquina universal consistem em sequências dos símbolos D, A, C, R, L e ;. Além disso, posicionamos estes seis símbolos nos F-quadrados da fita de $\mathcal{U}^{\mathcal{T}}$. E tais símbolos continuarão a aparecer apenas em F-quadrados durante o trabalho da máquina.

Quando definimos a versão primária das máquinas de Turing, havíamos mencionado que ela admitia entradas não-vazias, onde se intercalavam, entre os símbolos, alguns espaços vazios. Pois que, devido ao espaço vazio, segundo a definição algébrica de máquina de Turing que demos por último, faremos uma adaptação.

No lugar do ␣, usaremos, dentro do alfabeto de entrada, o símbolo $\widehat{␣}$. Ainda, o alfabeto de entrada de $\mathcal{U}^{\mathcal{T}}$ poderia contar com o símbolo ə, para ser colocado no início da fita. Mas optaremos por deixar este lugar para inserir, inicialmente, dois símbolos $\widehat{␣}$. Mais à frente explicaremos o porquê. E podemos, também, adicionar o símbolo ::, o qual marcará o fim da cadeia de entrada.

Sendo $\Sigma_{\mathcal{U}^{\mathcal{T}}}$ tal alfabeto de entrada, obviamente nem todas as entradas nos serão proveitosas. Deter-nos-emos apenas nas cadeias que apresentarem a forma de uma descrição padrão junto das especificidades como recentemente comentamos.

A seguir, tomamos um exemplo de máquina $\mathcal{M}^{\mathcal{T}}$ e o que seria a cadeia dela a estar na entrada da máquina universal. Trata-se de uma máquina que computa a sequência relativa ao número $\left(0,\overline{10}\right)_2$.

O alfabeto da fita desta máquina $\mathcal{M}^{\mathcal{T}}$ consiste dos símbolos $\alpha_0 = ␣$, $\alpha_1 = 0$, $\alpha_2 = 1$ e $\alpha_3 =$ ə. A descrição tabular dela corresponde ao que esquematizamos logo abaixo.

Tabela 53 – Descrição da máquina que computa $\left(0,\overline{10}\right)_2$

Estado	Símbolo	Operações	Estado final
$\mathfrak{q}_0$	Qualquer	Pə, R	$\mathfrak{q}_1$
$\mathfrak{q}_1$	Qualquer	Pə, R	$\mathfrak{q}_2$
$\mathfrak{q}_2$	Qualquer	P1, R	$\mathfrak{q}_3$
$\mathfrak{q}_3$	Qualquer	P␣, R	$\mathfrak{q}_4$
$\mathfrak{q}_4$	Qualquer	P0, R	$\mathfrak{q}_5$
$\mathfrak{q}_5$	Qualquer	P␣, R	$\mathfrak{q}_2$

Fonte: Autoria própria

A descrição algébrica ordenada da máquina em questão consta abaixo.

$$\mathfrak{q}_0\alpha_0\alpha_3\mathrm{R}\mathfrak{q}_1;\ \mathfrak{q}_0\alpha_1\alpha_3\mathrm{R}\mathfrak{q}_1;\ \mathfrak{q}_0\alpha_2\alpha_3\mathrm{R}\mathfrak{q}_1;\ \mathfrak{q}_0\alpha_3\alpha_3\mathrm{R}\mathfrak{q}_1;$$

$$\vdots$$

$$\mathfrak{q}_5\alpha_0\alpha_0\mathrm{R}\mathfrak{q}_2;\ \mathfrak{q}_5\alpha_1\alpha_0\mathrm{R}\mathfrak{q}_2;\ \mathfrak{q}_5\alpha_2\alpha_0\mathrm{R}\mathfrak{q}_2;\ \mathfrak{q}_5\alpha_3\alpha_0\mathrm{R}\mathfrak{q}_2;$$

Em seguida, sua descrição padrão.

DADDCCCRDAA;DADCDCCCRDAA;

DADCCDCCCRDAA;DADCCCDCCCRDAA;

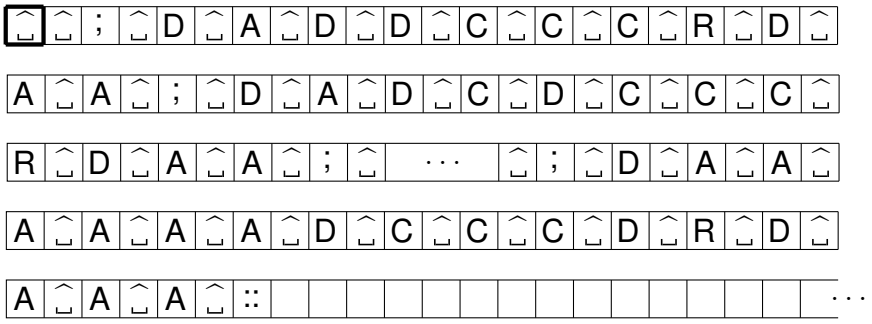
$$\vdots$$

DAAAAAADDRDAAA;DAAAAAADCDRDAAA;

DAAAAAADCCDRDAAA;DAAAAAADCCCDRDAAA;

Dado isso, na entrada da máquina universal, teríamos o que está representado na figura abaixo.

Figura 41 – Exemplo de entrada conveniente para a máquina de Turing universal



Fonte: Autor (2024)

Pois bem. Recordemos dois conceitos. Chamamos de *configuração* de um estágio a informação do estado atual junto do símbolo escaneado pela máquina. E chamamos de *configuração completa* de um estágio a informação da presente sequência de símbolos da fita, junto do quadrado escaneado e do estado atual.

Chamaremos de **instrução** a descrição de algum estado específico, junto de um símbolo escaneável, das operações que a máquina faz a partir destes, e do estado sucessor. Ou seja, se trata, no contexto da fita da máquina $\mathcal{U}^{\mathcal{T}}$, da informação contida entre dois símbolos ; seguidos.

Uma instrução é também o conteúdo de uma linha das tabelas descritivas das máquinas que vínhamos apresentando. E também pode ser considerada como um ponto no gráfico da função de transição.

Serão bastante utilizadas m-funções antes introduzidas. As m-funções:

- $\mathfrak{f}$ e $\mathfrak{g}$, que encontram, respectivamente, a primeira e a última ocorrência de dado símbolo;
- $\mathfrak{cpe}$, que compara o conteúdo de F-quadrados marcados por um par de E-símbolos e apaga tais marcadores em seguida;
- $\mathfrak{pe}$, que imprime um dado símbolo no fim da fita;
- $\mathfrak{e}$, que apaga os marcadores da fita;
- $\mathfrak{ce}$, que copia em ordem, para o fim da fita, os símbolos que possuem algum marcador escolhido, e em seguida apaga tais marcadores.

O papel da máquina universal consiste em, a partir da descrição padrão de uma máquina $\mathcal{M}^{\mathcal{T}}$ qualquer, escrever a sequência que ela computaria, imprimindo, para

isso, um F-símbolo por vez. Com essa finalidade, a máquina $\mathcal{U}^{\mathcal{T}}$ segue, basicamente, o algoritmo a seguir.

Para a $\mathcal{M}^{\mathcal{T}}$ de que a máquina universal estaria replicando a computação, há a certeza de que o estado inicial de tal máquina seria o chamado $\mathfrak{q}_0$, e que o primeiro símbolo escaneado seria o espaço em branco. Então ela imprime, nos primeiros F--quadrados após o fim da fita, a sequência de símbolos :, D, A e D, indicando que está na configuração com $\mathfrak{q}_0$ (DA) e o símbolo escaneado $\textvisiblespace$ (D).

A partir daí, ela vai começar um procedimento cíclico. O próximo passo é marcar essa configuração com um símbolo y, em cada um de seus F-quadrados. Em seguida, vai ser feito um teste com uma por uma das instruções, desde a última delas até a primeira. Tal teste consiste em: marcar a instrução, em seu primeiro E-quadrado (ao lado do símbolo ;), com um símbolo z; marcar sua configuração potencial (ou seja, seu estado atual e seu símbolo escaneável) com símbolos x; e comparar a configuração potencial da instrução com a configuração no final da fita.

A configuração no final de $\mathcal{U}^{\mathcal{T}}$ representa a configuração atual de $\mathcal{M}^{\mathcal{T}}$. Se não forem iguais, apagam-se os marcadores x e y, e marca-se novamente a configuração no final da fita (com o y), para que ela seja comparada com a configuração potencial presente na instrução anterior, pelo mesmo método.

Assim que duas configurações baterem, a máquina marca os quadrados correspondentes às operações da instrução com os símbolos u e o estado sucessor com os símbolos y. Também apaga todos os marcadores z da fita, e então fará algo que, no primeiro momento, não é percebido, mas que em próximas configurações completas fica visível.

Ocorre que, sempre que se imprimem as configurações completas ao final da fita de $\mathcal{U}^{\mathcal{T}}$, elas estão dispostas, nos F-quadrados, do seguinte modo. Inicia com o símbolo :. Há os símbolos D e C para representar os símbolos encontrados na fita de $\mathcal{M}^{\mathcal{T}}$ que precedem o símbolo escaneado. Há, então, os símbolos D e A para representar o estado atual. E, por fim, os símbolos D e C para representar o símbolo escaneado e os símbolos posteriores da fita de $\mathcal{M}^{\mathcal{T}}$.

Agora a máquina universal se dirige para a configuração completa no final de sua fita. O que ela faz, neste momento, é marcar com símbolos v os D e C precedentes da representação do estado atual, exceto o D e os C antecessores imediatos dela. Estes ela marca com os símbolos x. Ela deixa a representação do estado atual e símbolo escaneado sem marcadores e, então, marca os D e C posteriores com símbolos w. E, por fim, imprime um : no F-quadrado seguinte.

Depois disso, a cabeça da máquina universal andará para trás a fim de examinar as operações da instrução antes marcada com u. Se a operação é de imprimir o símbolo 0, isto é, possui a representação DC nos F-quadrados de $\mathcal{U}^{\mathcal{T}}$, então a cabeça vai para o fim da fita e imprime, respectivamente, 0 e : nos próximos F-quadrados.

Se a operação é de imprimir o símbolo 1, isto é, possui a representação DCC nos F-quadrados de $\mathcal{U}^{\mathcal{T}}$, então a cabeça vai para o fim da fita e imprime, respectivamente, 1 e : nos próximos F-quadrados.

Se a operação era de imprimir outro símbolo, então a cabeça não o imprimirá no fim da fita, pois só interessa à máquina universal registrar os F-símbolos computados por $\mathcal{M}^{\mathcal{T}}$. Depois disso, a cabeça de $\mathcal{U}^{\mathcal{T}}$ volta a andar para trás para examinar, desta vez, a operação de deslocamento que constava na instrução. Se ele for R ou L, a máquina vai agir de um jeito ou de outro.

Pois o próximo passo é copiar, para o final da fita, a próxima configuração completa em que $\mathcal{M}^{\mathcal{T}}$ se encontraria. De qualquer maneira, algumas coisas acontecerão com certeza. A cabeça apagará o marcador u presente na operação de deslocamento. A máquina copiará, primeiramente, todos os símbolos (da instrução) marcados com v para o final da fita, que eram os D e C que representavam os primeiros símbolos da antiga configuração completa. E também a máquina copiará, por último, todos os símbolos marcados com w para o fim da fita, que eram os D e C que representavam os símbolos da antiga configuração completa encontrados após o quadrado escaneado.

O que muda são os símbolos copiados para o fim da fita depois dos marcados com v e antes dos marcados com w. Pois, se a operação escaneada foi R, a máquina copiará para o fim da fita os símbolos marcados, respectivamente, por x, u e y. E se a operação escaneada foi L, a máquina copiará para o fim da fita os símbolos marcados, respectivamente, por y, x e u.

Afinal, se na máquina $\mathcal{M}^{\mathcal{T}}$ a cabeça se moveria para a direita, a próxima configuração completa teria o seu estado atual, o qual marcamos com y, imediatamente depois do símbolo recém-impresso, o qual marcamos com u. E se em $\mathcal{M}^{\mathcal{T}}$ a cabeça se moveria para a esquerda, a próxima configuração completa teria o seu estado atual imediatamente antes do antigo símbolo antecessor, o qual marcamos com x.

Dado isso, todos os marcadores são apagados e temos novamente uma configuração completa ao final da fita. Nela, marcaremos com y sua configuração (estado atual e símbolo escaneado de $\mathcal{M}^{\mathcal{T}}$) e o ciclo recomeça.

Aqui cabe observar uma coisa. Os símbolos $\widehat{\sqcup}$ só apareceram na fita durante a primeira vez que o ciclo acima referido estiver em ação. Pois, ao final dela, todos os marcadores serão apagados. Mas como eles estarão presentes nessa vez, cabe definirmos devidamente a função de transição para que eles não causem nenhum problema.

Seja $\delta_{\mathcal{U}^{\mathcal{T}}}$ a função de transição da máquina universal. Em verdade, só o que precisamos é afirmar que, para todo $\mathfrak{q}$ do conjunto de estados de $\mathcal{U}^{\mathcal{T}}$, e para algum $\mathfrak{q}'$ do mesmo conjunto, algum α do alfabeto de entrada, e algum $\mathsf{X} \in \{\mathsf{R}, \mathsf{L}\}$, temos que $\delta_{\mathcal{U}^{\mathcal{T}}}(\mathfrak{q}, \sqcup) = (\mathfrak{q}', \alpha, \mathsf{X})$ se, e somente se $\delta_{\mathcal{U}^{\mathcal{T}}}(\mathfrak{q}, \widehat{\sqcup}) = (\mathfrak{q}', \alpha, \mathsf{X})$.

Isso dá conta de que, no início de sua atividade, a máquina universal imprima

os dois ə do início da fita. Para essa finalidade é que lá estavam presentes os dois, $\widehat{\sqcup}$, garantindo a o mesmo limitador presente na computação das máquinas $\mathcal{M}^{\mathcal{T}}$.

Em verdade, esse recurso estabelecido na função de transição dá conta, em geral, de que a máquina universal aja do mesmo jeito ao escanear $\sqcup$ e $\widehat{\sqcup}$.

Mais uma coisa a comentarmos é que, da maneira como foi elaborada a computação dos F-símbolos da $\mathcal{M}^{\mathcal{T}}$ na máquina universal, há um problema. As máquinas cuja computação estamos replicando de fato não imprimem novos símbolos em cima dos F-quadrados já preenchidos. Só que elas podem, por ventura, voltar a algum desses F-quadrados, e em cima deles imprimirão o mesmo F-símbolo que já estava ali, seguindo as convenções de Sipser. Só que a máquina universal, no momento presente, não faz essa diferenciação. Então ela irá computar novamente o mesmo F-símbolo sem que ele realmente tenha sido impresso em um novo F-quadrado.

Para arrumarmos isso, se percebemos que o símbolo escaneado representado na configuração completa é o 0 ou o 1, não iremos marcar os quadrados correspondentes com y. Prevendo isso, vamos adicionar dois símbolos no alfabeto da fita de $\mathcal{U}^{\mathcal{T}}$. Trata-se de colocar nos marcadores x e y um "chapéu". Ou seja, os novos símbolos são $\widehat{\mathrm{x}}$ e $\widehat{\mathrm{y}}$.

Assim, tal configuração (que contém o F-símbolo), no fim da fita, será marcada com $\widehat{\mathrm{y}}$. A máquina universal fará o exame para encontrar qual é a instrução correspondente a essa configuração, e marcará sua configuração com os símbolos $\widehat{\mathrm{x}}$. Em seguida, os outros passos são semelhantes. A diferença é que, se adentramos neste presente caso, a máquina não imprimirá o F-símbolo ao lado da configuração completa como antes faria. Ela simplesmente irá levar para o fim da fita a próxima configuração completa.

Originalmente a máquina universal não contemplava isso. Ela fora descrita, usando as m-funções comentadas antes, com uma série de m-funções próprias. Fizemos, aqui, algumas modificações e acréscimos envolvendo as m-funções que descrevem a máquina universal para dar conta do problema acima, mas deixamos para detalhá-los no apêndice B.

Pode ser que a máquina $\mathcal{U}^{\mathcal{T}}$ receba como entrada a descrição de uma máquina de Turing que não seja da classe das máquinas $\mathcal{M}^{\mathcal{T}}$. Mas aconteça que tal máquina tenha instruções que lhe permitam reproduzir sua computação dentro da máquina universal, ou até entrar em conflito com ela, ou talvez seguindo sem problemas.

Imaginemos, pois, uma máquina que imprima 0 e 1 no que seriam os F-quadrados, mas que, por ventura, substitui tais F-símbolos por outros nestes próprios quadrados. Sendo assim, a reprodução de suas configurações completas esbarrará neste problema, dadas as descrições que fizemos da máquina universal.

Ou então, pensemos em uma máquina que deixe alguns F-quadrados com símbolos alternativos a 0 e 1 ao longo de sua computação. Tais símbolos não serão

registrados nas configurações completas da reprodução dentro da máquina universal.

E por isso, destacamos aqui o objetivo da presente máquina universal. Trata-se de reproduzir especificamente a computação das máquinas $\mathcal{M}^{\mathcal{T}}$. E isto ela fará. Ela não diferenciará as entradas que são descrições de uma máquina dessa classe das que não são. Ela continuará fazendo as transições como antes explicado.

Dito isso, não nos preocupemos a respeito de como ela vai agir perante entradas que não são do interesse expresso. Ela foi considerada para fazer o trabalho descrito diante das entradas convenientes. Vendo que ela cumpre este papel, sigamos em frente.

5 SOBRE DECIDIBILIDADE

Neste capítulo visamos explorar uma série de problema cuja resolução (ou não) é basilar para o trato com problemas em capítulos posteriores. Para isso, nos aprofundaremos no uso das linguagens e começaremos a trabalhar muito mais com a noção de decidibilidade.

5.1 ALGORITMOS E MÁQUINAS DE TURING

Agora, adentraremos no uso dos **algoritmos**, que são, basicamente, instruções simples para realizar uma tarefa. São receitas. Eles podem, no contexto da dissertação, ser definidos em termos de máquinas de Turing. Serão, em geral, o procedimento empregado por uma máquina de Turing. Algoritmos podem ser detalhados ao nível da chamada **descrição formal**, podem sê-lo ao nível da **descrição de implementação**, ou ainda, ao nível da **descrição de alto-nível**. Para as tarefas nas máquinas de Turing, estes são, respectivamente, os níveis mais baixo até o mais alto de abstração. São assim considerados por descreverem o algoritmo com cada vez menos detalhes e com mais generalidade.

No primeiro destes níveis, o da descrição formal, expomos os detalhes da função de transição, os estados da máquina, esse tipo de coisa. Sua descrição é algébrica e extensa. No nível da descrição de implementação falamos do movimento da cabeça e de como são armazenados os dados ao longo da fita, sem detalhar a função de transição e os estados da máquina. No terceiro nível focamos mais no que a máquina fará genericamente com sua computação, sem adentrar nas operações da cabeça e preenchimento detalhado da fita.

Tanto no segundo e terceiro níveis, utilizamos muito mais a linguagem natural para descrever. Enquanto que no primeiro nível a descrição é mais matemática. Já fornecemos algumas descrições formais de máquinas de Turing até agora na dissertação. E, na seção anterior, quando contamos como procede a máquina universal, podemos dizer que fornecemos uma descrição de implementação. Agora focaremos em descrições de alto-nível.

Até onde temos trabalhado, máquinas de Turing começam sempre com cadeias de entrada. E assim continuaremos. Iremos utilizar objetos para “serem lidos” por máquinas de Turing, mas isso porque os converteremos em uma cadeia específica correspondente. Diremos, para algum objeto O, que a sua representação como uma cadeia é a **codificação de** O, a qual denotamos por $\langle O \rangle$. Por exemplo, codificamos máquinas $\mathcal{M}^{\mathcal{T}}$, há pouco, em cadeias do alfabeto $\{\mathsf{A}, \mathsf{C}, \mathsf{D}, \mathsf{R}, \mathsf{L}, ;, :, ::, \widehat{\sqcup}\}$. Essa é uma

opção. Não quer dizer que é a única.

Quando temos vários objetos, $O_1, O_2, \ldots, O_k$, para colocar na entrada da máquina, o fazemos com uma única cadeia denotada $\langle O_1, O_2, \ldots, O_k \rangle$. Descreveremos os algoritmos a seguir com um passo a passo enumerado e finito. Colocamos o algoritmo entre aspas ao longo de parágrafos, escrevendo-o como que no segundo membro de uma igualdade, onde o primeiro membro é a máquina de Turing que ele estiver descrevendo.

Quando o algoritmo for voltado a computar a partir de codificações de objetos, há sempre, implicitamente, a verificação de se a cadeia de entrada realmente representa devidamente um objeto da forma desejada. Se ela verifica que ela assim representa, prossegue. E se ela verifica que ela não consegue representar, então rejeita tal entrada.

Abaixo damos um exemplo de como seria a descrição em alto-nível do algoritmo da máquina universal anteriormente comentada. A verificação implícita se limitará a respeito de a entrada codificar ou não uma máquina de Turing.

Exemplo 9. Sendo $\mathcal{U}^{\mathcal{T}}$ a máquina de Turing universal, temos que

$\mathcal{U}^{\mathcal{T}} =$ **"** Sobre uma entrada $\langle \mathcal{M}^{\mathcal{T}} \rangle$, onde $\mathcal{M}^{\mathcal{T}}$ é da classe das máquinas que mais cedo definimos, e a codificação é tal como já usamos:

1. Imprima a configuração inicial de $\mathcal{M}^{\mathcal{T}}$ ao final da fita.

2. Compare a configuração ao fim da fita com as configurações das instruções da entrada até encontrar a instrução que possua a mesma configuração do fim.

3. Imprima, ao lado da configuração completa ao fim da fita, o F-símbolo que a instrução ordenaria imprimir, a menos que o símbolo atual da configuração já seja um F-símbolo.

4. Imprima, ao fim da fita, a representação da configuração completa de acordo com o que a instrução encontrada ordenou a máquina $\mathcal{M}^{\mathcal{T}}$ a fazer.

5. Encontre e marque a configuração presente na configuração completa ao final da fita. E então, retorne ao terceiro passo do algoritmo. **"** □

Vendo como se dá a descrição de alto-nível de um algoritmo, passaremos a utilizá-las para resolver alguns problemas, primeiramente, quanto a linguagens.

5.2 AUTÔMATOS FINITOS

Para tratarmos de linguagens mais simples, inicialmente, apresentaremos os chamados **autômatos finitos**. Eles são como que máquinas de Turing cuja cabeça só anda para a direita na fita. Sua fita é finita, do tamanho exato da entrada que a máquina estiver lendo no momento. Seu papel não está em mudar os símbolos que estão nos quadrados, mas apenas os estados.

Uma diferença para as máquinas de Turing é que as cadeias de entrada dos autômatos finitos admitem o $\sqcup$ como um de seus símbolos.

Outra diferença é que, em vez de simplesmente um estado de aceitação, eles possuem um conjunto de estados de aceitação. Se, após escanear o último dos símbolos da cadeia de entrada, o autômato finito se encontrar em um estado de aceitação, dizemos que ele aceitou a entrada. Se não, ele não aceitou.

Os autômatos finitos podem, assim como as máquinas de Turing, ser determinísticos ou não-determinísticos. Abaixo, a definição algébrica do caso determinístico.

Definição 10. Sendo Q e Σ conjuntos finitos, chamamos de **autômato finito determinístico** toda 5-upla $(Q, \Sigma, \delta, \mathfrak{q}_0, F)$ tal que

1. Q é o conjunto de estados;
2. Σ é o alfabeto;
3. $\delta : Q \times \Sigma \to Q$ é a função de transição;
4. $\mathfrak{q}_0 \in Q$ é o estado inicial;
5. e $F \subseteq Q$ é o conjunto de estados de aceitação.

Também chamados de AFD, os autômatos finitos determinísticos podem ter desde nenhum até todos os seus estados como estados de aceitação.

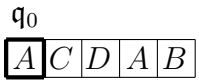
Exemplo 10. Sejam $Q = \{\mathfrak{q}_0, \mathfrak{q}_1, \mathfrak{q}_2, \mathfrak{q}_3\}$, $\Sigma = \{A, B, C, D\}$ e $F = \{\mathfrak{q}_1, \mathfrak{q}_3\}$. Seja $\delta : Q \times \Sigma \to Q$ a função tal que, para todo $\alpha \in \Sigma$:

$\delta(\mathfrak{q}_0, \alpha) = \delta(\mathfrak{q}_1, D) = \delta(\mathfrak{q}_2, B) = \delta(\mathfrak{q}_2, C) = \delta(\mathfrak{q}_3, D) = \mathfrak{q}_3$;
$\delta(\mathfrak{q}_1, A) = \delta(\mathfrak{q}_1, C) = \delta(\mathfrak{q}_2, A) = \delta(\mathfrak{q}_3, A) = \mathfrak{q}_2$;
$\delta(\mathfrak{q}_1, B) = \delta(\mathfrak{q}_2, D) = \delta(\mathfrak{q}_3, B) = \mathfrak{q}_1$;
$\delta(\mathfrak{q}_3, C) = \mathfrak{q}_0$.

Seja então o AFD $X = (Q, \Sigma, \delta, \mathfrak{q}_0, F)$. Qual será o veredito deste AFD diante da entrada $ACDAB$?

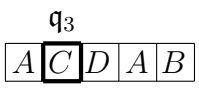
Começando no estado $\mathfrak{q}_0$, tal AFD, pela função de transição, vai sempre para o estado $\mathfrak{q}_3$, não importa qual seja o símbolo escaneado. E, com isso, o AFD se dirige para o próximo símbolo, que é C.

Figura 42 – Computação do exemplo de AFD com entrada $ACDAB$ (parte 1)



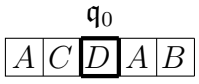
Fonte: Autor (2024)

Figura 43 – Computação do exemplo de AFD com entrada $ACDAB$ (parte 2)



Fonte: Autor (2024)

Figura 44 – Computação do exemplo de AFD com entrada $ACDAB$ (parte 3)

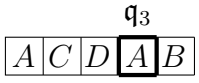


Fonte: Autor (2024)

No estado $\mathfrak{q}_3$ escaneando C, o AFD vai para os estado $\mathfrak{q}_0$ e próximo símbolo.

Agora o AFD escaneia o símbolo D no estado $\mathfrak{q}_0$. Passa para o símbolo, A, no estado $\mathfrak{q}_3$.

Figura 45 – Computação do exemplo de AFD com entrada $ACDAB$ (parte 4)



Fonte: Autor (2024)

No estado $\mathfrak{q}_3$, escaneando A, nosso AFD passa para o próximo símbolo e para o estado $\mathfrak{q}_2$.

Figura 46 – Computação do exemplo de AFD com entrada $ACDAB$ (parte 5)

Fonte: Autor (2024)

E, por fim, no estado $\mathfrak{q}_2$, com o símbolo B, passa para o estado $\mathfrak{q}_3$, e o autômato finaliza sua atividade.

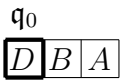
Figura 47 – Computação do exemplo de AFD com entrada $ACDAB$ (parte 6)

Fonte: Autor (2024)

Como o estado $\mathfrak{q}_3$ pertence a F, segue que X aceita a entrada $ACDAB$.

Agora, se fosse, a entrada DBA, como seria o veredito do nosso autômato finito determinístico?

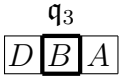
Figura 48 – Computação do exemplo de AFD com entrada DBA (parte 1)



Fonte: Autor (2024)

Começando pelo estado $\mathfrak{q}_0$, ao ler o símbolo D, X passa para o estado $\mathfrak{q}_3$. E anda para a direita.

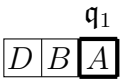
Figura 49 – Computação do exemplo de AFD com entrada DBA (parte 2)



Fonte: Autor (2024)

No estado $\mathfrak{q}_3$, lendo o símbolo B, o próximo estado é $\mathfrak{q}_1$. E vai para a direita.

Figura 50 – Computação do exemplo de AFD com entrada DBA (parte 3)



Fonte: Autor (2024)

E, por fim, no estado $\mathfrak{q}_1$, com o símbolo A, o AFD passa para o estado $\mathfrak{q}_2$, e termina sua atividade.

Figura 51 – Computação do exemplo de AFD com entrada DBA (parte 4)

Fonte: Autor (2024)

Como $\mathfrak{q}_2$ não pertence a F, segue que X não aceita a entrada DBA. □

Tal como nas máquinas de Turing, mencionamos anteriormente que também há uma versão não-determinística de nosso objeto. São os autômatos finitos não-determinísticos, ou AFN's. No entanto, não nos deteremos neles.

Assim como com máquinas de Turing, sendo B um autômato qualquer, há sempre uma linguagem sendo reconhecida. É a linguagem que consiste de todas as entradas de Σ^* a partir das quais chegamos a um estado de aceitação. E denotamos por $L(B)$ a linguagem que tal autômato reconhece.

É interessante vermos que sempre há alguma máquina de Turing "equivalente" a um AFD, no sentido de reconhecerem a mesma linguagem. Em verdade há uma

exceção a isto, que é quando o alfabeto do AFD admite $\lrcorner$ como um de seus símbolos. Mas mostraremos, abaixo, como se dá a relação.

Proposição 5.2.1. *Para todo AFD cujo alfabeto não admita o símbolo $\lrcorner$, há sempre alguma máquina de Turing que reconhece a mesma linguagem.*

Demonstração. Seja $B = (Q, \Sigma, \delta, \mathfrak{q}_0, F)$ um autômato finito determinístico, tal que $\lrcorner \notin \Sigma$.

Será montada a máquina de Turing $\mathcal{M}_B$ que vai reconhecer a linguagem de B. Definiremos $\mathcal{M}_B = (Q_0, \Sigma, \Sigma \cup \{\lrcorner\}, \delta_0, \mathfrak{q}_0, \mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}})$.

Para isso, sejam $Q \subsetneq Q_0$ e $\mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}} \notin Q$. Se para $\alpha \in \Sigma$ e $\mathfrak{q} \in Q$ tivermos $\delta(\mathfrak{q}, \alpha) = \mathfrak{q}'$, então $\delta_0(\mathfrak{q}, \alpha) = (\mathfrak{q}', \alpha, \mathsf{R})$.

Seja $i \in \mathbb{N}$. Se tivermos $\mathfrak{q}_i \in Q - F$, então $\delta_0(\mathfrak{q}_i, \lrcorner) = \big(\mathfrak{q}_{\text{rejeita}}, \lrcorner, \mathsf{R}\big)$. E se $\mathfrak{q}_i \in F$ teremos $\delta_0(\mathfrak{q}_i, \lrcorner) = (\mathfrak{q}_{\text{aceita}}, \lrcorner, \mathsf{R})$.

Tal máquina de Turing reconhece a linguagem de B. Afinal, quando chega ao final da fita, se B iria para um estado de F no fim da entrada, segue que este mesmo estado, em $\mathcal{M}_B$, levará ao estado de aceitação quando chegar na parte vazia da fita. E se B iria para um estado que não de F ao fim da fita, segue que o mesmo estado, em $\mathcal{M}_B$, levará ao estado de rejeição na parte vazia da fita. ■

Observação 5. Seja $B = (Q, \Sigma, \delta, \mathfrak{q}_0, F)$ um AFD cuja linguagem admita cadeias com o símbolo $\lrcorner$. Ele não tem uma máquina de Turing "equivalente" a ele, mas ainda assim ele ainda guardará uma correspondência com outros AFD's.

Imaginemos, pois, um AFD $B' = (Q, \Sigma_1, \delta_1, \mathfrak{q}_0, F)$. Este será tal que, sendo $\beta \notin \Sigma$ um símbolo qualquer, então $\Sigma_1 = (\Sigma \cup \{\beta\}) - \{\lrcorner\}$; e, se para $\alpha \in \Sigma - \{\lrcorner\}$ e $\mathfrak{q} \in Q$, e sabendo que $\delta(\mathfrak{q}, \alpha) = \mathfrak{q}'$ e $\delta(\mathfrak{q}, \lrcorner) = \overline{\mathfrak{q}}$, teremos que

$$\delta_1(\mathfrak{q}, \gamma) = \begin{cases} \mathfrak{q}', & \text{se } \gamma = \alpha, \\ \overline{\mathfrak{q}}, & \text{se } \gamma = \beta \end{cases}.$$

Acontece que a linguagem deste autômato B' é, basicamente, a linguagem do autômato B, substituindo-se todos os símbolos $\lrcorner$ por β. E, por sua vez, B' é "equivalente" a alguma máquina de Turing, como demonstramos recentemente.

5.3 LINGUAGENS DECIDÍVEIS

Apresentados os autômatos, iremos agora para o roteiro de utilizar linguagens para resolução de certos problemas. Começamos com o chamado ***accepting problem*** ou **problema de aceitação** (para AFD's), que trata se é possível saber se um dado AFD aceita uma determinada cadeia.

Seja Σ um alfabeto capaz de fazer as codificações que usaremos na sequência. Uma maneira de utilizar linguagens para resolver tal problema seria adotando a linguagem

$$\mathsf{A_{AFD}} = \{\langle B, w\rangle \in \Sigma^*;\ B \text{ é um AFD que aceita a cadeia de entrada } w\}.$$

Pois que, o *accepting problem* é equivalente ao problema de testar se a cadeia $\langle B, w\rangle$ aceita é ou não um membro da linguagem $\mathsf{A_{AFD}}$.

E, para resolver este último problema, basta descobrirmos se a linguagem $\mathsf{A_{AFD}}$ é decidível, já em uma máquina de Turing padrão mesmo. Afinal, se ela for decidível, isto significa que podemos fazer uma "receita" através da qual aceitamos todas as cadeias que o autômato aceita, e rejeitar, com a máquina decisora, todas as outras.

Ou seja, se por algum algoritmo mostrarmos exatamente quais cadeias $\langle B, w\rangle$ estão em $\mathsf{A_{AFD}}$, teremos resolvido o problema de aceitação. E enunciamos a conclusão desta questão pelo teorema abaixo.

Teorema 5.3.1. *A linguagem A_{AFD} é decidível.*

Demonstração. Definamos que a codificação de um AFD com uma cadeia de entrada, como usaremos aqui, apresenta a descrição da 5-upla que define o AFD, e também a cadeia de entrada presente.

Para tal, apresentamos a máquina de Turing descrita abaixo.

$\mathcal{M}$ = **"** Sobre a entrada $\langle B, w\rangle$, onde B é um AFD e w é uma cadeia de entrada do seu alfabeto:

1. Simule o funcionamento de B sobre a entrada w.
2. Se a simulação termina no que seria um estado de aceitação de B, então aceite. Se ela termina em um estado que não de aceitação, rejeite.**"**

Podemos verificar como é perceptível que tal máquina realmente dá conta do trabalho. Comecemos pelo fato de que a codificação de B com w vai trazer a informação da definição formal de B.

A primeira coisa que $\mathcal{M}$ faria é checar se sua cadeia de entrada realmente está na forma de uma codificação como requerida. Se não estiver, a máquina a rejeitará.

A máquina de Turing descrita no algoritmo pode agir de modo a representar cada símbolo do alfabeto de B como, por exemplo, um primeiro sendo D, um segundo sendo DC, um terceiro sendo DCC, e assim por diante; representar os estados de B que não sejam de aceitação como um primeiro sendo DA, um segundo sendo DAA, e assim por diante; e representar os estados de aceitação de B como um primeiro

sendo DE, um segundo sendo DEE, e assim por diante. E ir simulando a operação de B conforme as diretrizes da função de transição do AFD.

Temos que $\mathcal{M}$ manterá o registro do estado atual e da posição atual de B sobre a cadeia w. Após o último dos símbolos da cadeia indicados na codificação da cadeia w, a máquina de Turing irá para seu estado de aceitação se o último estado de B fosse representado com um D terminando com uma letra E. Se não fosse assim, iria para o estado de rejeição. ■

Estamos cogitando dentro da noção de que as máquinas de Turing apresentam uma maneira de agir suficientemente eficaz para resolver problemas que seriam solucionados por um conjunto finito de passos delimitados, isto é, por um algoritmo. Não estamos lidando aqui a fundo com a questão do tempo que levaria, mas sim com o fato de que o conseguiria em algum momento.

Acontece que esta concepção de que tais problemas solucionáveis por algoritmo encontram uma solução equivalente, mediante linguagens, utilizando as máquinas de Turing, é a postura geralmente adotada. Ela recebe o nome de **tese de Church-Turing**. Church por causa de Alonzo Church, que procurou definir "algoritmo" por um sistema de notações chamado λ-cálculo. E Alan Turing, pela definição através de suas máquinas de Turing.

As definições de Church e de Turing foram demonstradas equivalentes pelo próprio Turing. Por meio de ambas se procura formalizar a concepção intuitiva do que seriam os algoritmos. Devido ao natural limite da capacidade humana de raciocínio, é visível que, se há problemas que demandam certas condições ilimitadas para serem resolvidos, então a solução deles ordinariamente não estará ao nosso alcance.

Utilizando das ferramentas que dispomos para distinguir quais são esses problemas e quais não são, podemos, a partir disso, focar os nossos esforços em tarefas produtivas. Ao descobrirmos que a linguagem $\mathsf{A_{AFD}}$ é decidível, já sabemos algo das possibilidades que temos para estudar a questão dos AFD's.

Poderíamos tratar de outros problemas neste contexto dos autômatos. Um deles é o de **vacuidade** para a linguagem de um autômato finito. Trata-se de testar se, dado um autômato finito determinístico, há de fato alguma cadeia de entrada que ele aceite. Isso fica traduzido a partir da linguagem

$$\mathsf{V_{AFD}} = \{\langle B\rangle \in \Sigma^*;\ A \text{ é um AFD e } L(A) = \varnothing\}.$$

Outro problema seria o de determinar se dois AFD's reconhecem a mesma linguagem. Se ele é decidível ou não, o abordaríamos a partir da linguagem

$$\mathsf{EQ_{AFD}} = \{\langle B_1, B_2\rangle \in \Sigma^*;\ B_1 \text{ e } B_2 \text{ são AFD's e } L(B_1) = L(B_2)\}.$$

Em ambos os casos conseguiríamos demonstrar que sim, as linguagens, e consequentemente os problemas, são decidíveis. No entanto, o foco neste trabalho será colocado sim sobre os problemas correlacionados a estes, mas no universo das próprias máquinas de Turing.

Antes de partirmos para tais problemas, terminaremos esta seção com um teorema sobre equivalência a respeito de decidibilidade.

Primeiramente, tomemos a notação de que, para um conjunto X, seja $\overline{X}$ o conjunto complementar de X. Quando estivermos falando de um conjunto X que seja uma linguagem a partir de um alfabeto Σ, então $\overline{X} = \Sigma^* - X$.

Definição 11. Uma linguagem A é **co-Turing-reconhecível** se, e somente se $\overline{A}$ é uma linguagem Turing-reconhecível.

Para preparar o teorema, demonstremos um resultado anterior. Ele é aqui trazido devido ao fato de que, à primeira vista, mesmo que exista uma máquina de Turing que reconheça uma linguagem $A \subset \Sigma^*$, não sabemos se existe uma máquina assim que admita como seu alfabeto de entrada o alfabeto Σ.

Poderíamos pensar no caso em que A não utiliza todos os símbolos de Σ para compor suas cadeias. Nesta situação, poderia haver uma máquina de Turing que reconhece A e cujo alfabeto de entrada está contido em Σ, sendo, no entanto, distinto dele. E com isso, acabássemos pensando que não necessariamente há uma máquina que reconheça nossa linguagem e cujo alfabeto de entrada seja Σ.

Ou então, nos depararmos com o caso em que há uma máquina de Turing que reconhece A e cujo alfabeto de entrada não está contido em Σ. Nesta situação, tal alfabeto contém um subconjunto de Σ com o qual já é possível obter a linguagem A, e, novamente, pensássemos não existir certeza de que houvesse uma máquina com alfabeto de entrada Σ e cuja linguagem é A.

Lema 5.3.2. *Seja A uma linguagem a partir do alfabeto Σ. Se A é Turing-reconhecível, então existe alguma máquina de Turing que a reconhece e cujo alfabeto de entrada é Σ.*

Demonstração. Deixemos livre o alfabeto da fita. Uma máquina de Turing qualquer que reconhecesse esta linguagem, o faria devido à sua função de transição. Há dois diferentes casos em que o alfabeto de entrada de uma máquina cuja linguagem é A não é Σ: **i)** quando ele é um alfabeto $\Sigma' \subsetneq \Sigma$; e **ii)** quando ele é um alfabeto $\Sigma'' \supsetneq \Sigma'$ tal que $\Sigma'' \nsubseteq \Sigma$, e onde $A \subset (\Sigma')^*$ e $\Sigma' \subseteq \Sigma$.

Provaremos que, em qualquer um destes casos, ainda conseguimos montar uma máquina de Turing nas condições exigidas.

i) Tenhamos um alfabeto $\Sigma' = \{\alpha_1, \alpha_2, \ldots, \alpha_{m'}\}$ tal que $A \subset (\Sigma')^* \subsetneq \Sigma^*$. Seja $\mathcal{M}' = \left(Q', \Sigma', \Gamma', \delta', \mathfrak{q}'_0, \mathfrak{q}'_{\text{aceita}}, \mathfrak{q}'_{\text{rejeita}}\right)$ a presente máquina que reconhece A.

Montaremos a máquina $\mathcal{M}_1$, com alfabeto de entrada Σ, para reconhecer A. Ela terá um conjunto de estados ampliado Q_1, o qual contém Q'. Terá um alfabeto de entrada Γ_1 que contém Γ. Terá uma função de transição δ_1, na qual manteremos as mesmas transições da máquina anterior quando houverem as mesmas configurações. Seu estado inicial será um dos novos estados, o qual chamaremos de $\mathfrak{q}_0^1$. E seus estados de parada serão os mesmos de $\mathcal{M}'$.

Ela, inicialmente, fará uma varredura sobre a cadeia de entrada. Se encontrar um símbolo pertencente a $\Sigma - \Sigma'$ nela, rejeita a entrada. Se não encontrar, retorna para o começo da entrada e computa do jeito que $\mathcal{M}'$ faria.

Para tal, na configuração inicial, se o primeiro símbolo da fita não pertencer a $\Sigma - \Sigma'$, ela vai substitui-lo por um símbolo correspondente. Considerando $\alpha_0 = \text{␣}$ e sendo $\Gamma' = \{\alpha_0, \alpha_1, \ldots, \alpha_{M'-1}\}$, vamos utilizar de símbolos $\alpha_{M'}$, $\alpha_{M'+1}$, $\ldots$, $\alpha_{M'+m'}$ que não estivessem em Γ'. Tendo isso em mente, o nosso alfabeto da fita será $\Gamma_1 = \Gamma' \cup \{\alpha_{M'}, \alpha_{M'+1}, \ldots, \alpha_{M'+m'}\}$.

Seja $i = 0, 1, 2, \ldots, m'$. Teremos que $\delta_1\left(\mathfrak{q}_0^1, \alpha_i\right) = \left(\mathfrak{q}_1^1, \alpha_{M'+i}, \mathsf{R}\right)$. Na nossa transição, faremos a máquina, com o estado $\mathfrak{q}_1^1$, percorrer a cadeia de entrada sem mudar os símbolos, até que encontre o primeiro ␣. Se, antes disso acontecer, a máquina encontrar um símbolo de $\Sigma - \Sigma'$, ela vai para o estado de rejeição. Também vai para o estado de rejeição se, na configuração inicial, o símbolo pertencia a $\Sigma - \Sigma'$.

Quando durante o estado $\mathfrak{q}_1^1$ a máquina encontrar ␣, ela mudará para o estado $\mathfrak{q}_2^1$, sem mudar o símbolo ␣. No estado atual, a máquina vai andar para a esquerda.

Novamente, não vai haver mudança de símbolos, até ela encontrar um símbolo $\alpha_{M'+i}$. Quando o fizer, é porque chegou no início da fita. Neste quadrado, ela vai trocar o símbolo de volta para α_i, vai mudar para o estado $\mathfrak{q}_0'$ e andará para a esquerda (continuando, então, no quadrado inicial).

Tenha então, nossa máquina, percorrido todos esses passos sem ser levada à rejeição. Agora, como as transições de $\mathcal{M}_1$ são as mesmas de $\mathcal{M}'$ para as mesmas configurações, e a fita verificada só possui símbolos de Σ' e ␣ (o qual obviamente também estava na máquina $\mathcal{M}'$), segue que a linguagem que $\mathcal{M}_1$ reconhecer será a mesma de $\mathcal{M}'$.

ii) Chamemos de $\mathcal{M}''$ a máquina que, neste caso, reconhece A. Ela possui o alfabeto de entrada Σ'' antes descrito.

Mas já que $\mathcal{M}''$ aceita somente cadeias de $A \subset (\Sigma')^* \subset (\Sigma'')^*$, podemos montar uma máquina de Turing $\mathcal{M}_2$, com mesmo alfabeto da fita que $\mathcal{M}''$, mesmo conjunto de estados, mesmas transições também, mesmo estado inicial e estados de parada, mas cujo alfabeto de entrada seja Σ'.

A nova máquina ainda reconhecerá A. Ora, $\Sigma' \subseteq \Sigma$. Se for o caso em que $\Sigma' = \Sigma$, chegamos ao nosso objetivo. Se for o caso em que $\Sigma' \subsetneq \Sigma$, basta aplicarmos o mesmo procedimento que aplicamos no item **ii)** à máquina $\mathcal{M}_2$, e também chegaremos ao nosso objetivo. ■

Teorema 5.3.3. *Uma linguagem é decidível se, e somente se ela é Turing-reconhecível e co-Turing-reconhecível.*

Demonstração. Provemos primeiro que, se uma linguagem é decidível, então ela é Turing-reconhecível e co-Turing-reconhecível.

Se a linguagem A é decidível, significa que há uma máquina $\mathcal{M}$ que aceita suas cadeias e rejeita as cadeias fora dela. Então $\mathcal{M}$ reconhece A. Seja Σ o alfabeto de entrada de $\mathcal{M}$. Podemos construir a máquina $\mathcal{M}'$ como descrita abaixo.

$\mathcal{M}' =$ **"** Sobre a entrada w, onde w é uma cadeia sobre o alfabeto Σ:

1. Emule $\mathcal{M}$ sobre a entrada w.
2. Se a máquina $\mathcal{M}$ aceita, então rejeite. Se $\mathcal{M}$ rejeita, então aceite.**"**

Tal máquina acima é um decisor da linguagem $\overline{A}$. E, sendo assim ela reconhece a $\overline{A}$, ou seja, A é co-Turing-reconhecível.

Estando a primeira parte provada, agora demonstraremos que, se uma linguagem é Turing-reconhecível e co-Turing-reconhecível, então ela é decidível.

Seja, então, $\overline{A} = \Sigma^* - A$. Partimos da hipótese de que as linguagens A e $\overline{A}$ são Turing-reconhecíveis. Assim, pelo lema 5.3.2, existem uma máquina $\mathcal{M}_1$ que reconhece A e uma máquina $\mathcal{M}_2$ que reconhece $\overline{A}$, ambas com alfabeto de entrada Σ.

Sejam $\{\mathfrak{q}'_0, \mathfrak{q}'_1, \ldots, \mathfrak{q}'_{n_1}\}$ o conjunto de estados de $\mathcal{M}_1$ e $\{\mathfrak{q}''_0, \mathfrak{q}''_1, \ldots, \mathfrak{q}''_{n_2}\}$ o conjunto de estados de $\mathcal{M}_2$. E também $\mathfrak{q}'_{\text{aceita}}$ e $\mathfrak{q}'_{\text{rejeita}}$ os estados de aceitação e de rejeição de $\mathcal{M}_1$, e $\mathfrak{q}''_{\text{aceita}}$ e $\mathfrak{q}''_{\text{rejeita}}$ os estados de aceitação e de rejeição de $\mathcal{M}_2$.

Sejam δ_1 a função de transição de $\mathcal{M}_1$ e δ_2 a função de transição de $\mathcal{M}_2$. Vamos montar uma máquina $\mathcal{M}$ de duas fitas com mesmo alfabeto de entrada que $\mathcal{M}_1$ e $\mathcal{M}_2$ e cujo alfabeto da fita seja a união dos alfabetos da fita de $\mathcal{M}_1$ e $\mathcal{M}_2$. Sejam $\mathfrak{q}_{\text{aceita}}$ e $\mathfrak{q}_{\text{rejeita}}$ seus estados de aceitação e de rejeição respectivamente. Seja $\mathfrak{q}_0$ seu estado inicial e seja δ sua função de transição.

Tal máquina terá o seguinte trabalho. Ela recebe a entrada, copia-a para suas duas fitas, e então, na primeira fita age como agiria $\mathcal{M}_1$, enquanto que na segunda fita age como agiria $\mathcal{M}_2$. Se $\mathcal{M}_1$ aceitasse uma entrada w, então $\mathcal{M}$ a aceita. E se $\mathcal{M}_2$ aceitasse w, então $\mathcal{M}$ a rejeita. E só existem esses dois cenários. Pois uma entrada ou está em A, e por isso é aceita por $\mathcal{M}_1$, ou está em $\overline{A}$, e por isso é aceita por $\mathcal{M}_2$.

Também convém a programarmos para que, quando $\mathcal{M}_1$ rejeitasse w, então $\mathcal{M}$ rejeitasse. E quando $\mathcal{M}_2$ rejeitasse w, então $\mathcal{M}$ aceitasse. Pois se simplesmente ignorássemos as transições de rejeição e deixássemos a computação continuar, poderia vir a acontecer de uma sucessão futura da máquina que teria rejeitado w nos levasse ao estado de aceitação. Mostremos, pois, como nossa máquina funcionará.

Para $\beta' \neq \sqcup$ qualquer e β'' qualquer, teremos $\delta(\mathfrak{q}_0, \beta', \beta'') = (\mathfrak{q}_1, \beta', \beta'', \mathsf{R}, \mathsf{R})$, e teremos $\delta(\mathfrak{q}_0, \sqcup, \beta'') = (\mathfrak{q}_3, \sqcup, \beta'', \mathsf{L}, \mathsf{L})$. Teremos $\delta(\mathfrak{q}_1, \beta', \beta'') = (\mathfrak{q}_1, \beta', \beta'', \mathsf{R}, \mathsf{R})$ e $\delta(\mathfrak{q}_1, \sqcup, \beta'') = (\mathfrak{q}_2, \sqcup, \beta'', \mathsf{L}, \mathsf{L})$. Teremos, ainda, $\delta(\mathfrak{q}_2, \beta', \sqcup) = (\mathfrak{q}_2, \beta', \beta', \mathsf{L}, \mathsf{L})$ e, para $\beta \neq \sqcup$, $\delta(\mathfrak{q}_2, \beta', \beta) = (\mathfrak{q}_3, \beta', \beta, \mathsf{L}, \mathsf{L})$. Com isso, copiamos a entrada para a segunda fita, e começaremos a replicar as computações de $\mathcal{M}_1$ e $\mathcal{M}_2$ sobre w.

Para isso, teremos $\mathfrak{q}_3 = (\mathfrak{q}'_0, \mathfrak{q}''_0)$. Sejam $i, j \in \mathbb{N}$, $0 \leq i \leq n_1$, $0 \leq j \leq n_2$, sendo $i \notin \{i^{@}, i^{\#}\}$ se $\mathfrak{q}'_{i^{@}} = \mathfrak{q}'_{\text{aceita}}$ e $\mathfrak{q}'_{i^{\#}} = \mathfrak{q}'_{\text{rejeita}}$, e sendo $j \notin \{j^{@}, j^{\#}\}$ se $\mathfrak{q}''_{j^{@}} = \mathfrak{q}''_{\text{aceita}}$ e $\mathfrak{q}''_{j^{\#}} = \mathfrak{q}''_{\text{rejeita}}$. Se, para $\mathsf{X}_1, \mathsf{X}_2 \in \{\mathsf{R}, \mathsf{L}\}$, $\delta_1(\alpha', \mathfrak{q}'_i) = (\alpha^*, \mathfrak{q}^*, \mathsf{X}_1)$ e $\delta_2(\alpha'', \mathfrak{q}''_j) = (\alpha^{**}, \mathfrak{q}^{**}, \mathsf{X}_2)$, então

$$\delta\left(\left(\mathfrak{q}'_i, \mathfrak{q}''_j\right), \alpha', \alpha''\right) = \left(\left(\mathfrak{q}^*, \mathfrak{q}^{**}\right), \alpha^*, \alpha^{**}, \mathsf{X}_1, \mathsf{X}_2\right).$$

Enquanto que, por fim,

$$\delta\left(\left(\mathfrak{q}'_{\text{aceita}}, \mathfrak{q}''_j\right), \alpha', \alpha''\right) = \delta\left(\left(\mathfrak{q}'_{\text{aceita}}, \mathfrak{q}''_{\text{rejeita}}\right), \alpha', \alpha''\right) = \left(\mathfrak{q}_{\text{aceita}}, \alpha', \alpha'', \mathsf{R}, \mathsf{R}\right),$$

$$\delta\left(\left(\mathfrak{q}'_i, \mathfrak{q}''_{\text{aceita}}\right), \alpha', \alpha''\right) = \delta\left(\left(\mathfrak{q}'_{\text{rejeita}}, \mathfrak{q}''_{\text{aceita}}\right), \alpha', \alpha''\right) = \left(\mathfrak{q}_{\text{rejeita}}, \alpha', \alpha'', \mathsf{R}, \mathsf{R}\right),$$

$$\delta\left(\left(\mathfrak{q}'_{\text{rejeita}}, \mathfrak{q}''_j\right), \alpha', \alpha''\right) = \left(\mathfrak{q}_{\text{rejeita}}, \alpha', \alpha'', \mathsf{R}, \mathsf{R}\right),$$

e

$$\delta\left(\left(\mathfrak{q}'_i, \mathfrak{q}''_{\text{rejeita}}\right), \alpha', \alpha''\right) = \left(\mathfrak{q}_{\text{aceita}}, \alpha', \alpha'', \mathsf{R}, \mathsf{R}\right).$$

As transições que não foram especificadas ficam livres. Tais como aquelas que envolvessem símbolos que só se encontravam, anteriormente, no alfabeto da fita de $\mathcal{M}_1$, considerados agora sobre a segunda fita; ou que envolvessem símbolos que só se encontravam, anteriormente, no alfabeto da fita de $\mathcal{M}_2$, considerados agora sobre a primeira fita.

Vendo que a máquina multifita $\mathcal{M}$ cumpre o papel que descrevemos acima, segue que ela é um decisor para a linguagem A. E, pelo teorema 4.1.1 e sua demonstração, sabemos que existe uma máquina de Turing de uma fita só que aceita as mesmas entradas que $\mathcal{M}$ e rejeita as mesmas entradas que $\mathcal{M}$.

Disso concluímos que a linguagem A é decidível. E a segunda parte está provada. ■

5.4 *HALTING PROBLEM*

Até agora vimos exemplos de linguagens decidíveis, as quais nos mostram que certos problemas são solucionáveis por algum número finito de passos. Eis que ressurge, então, a pergunta do *Entscheidungsproblem*: "existe algum problema para o qual não há, com um número finito de passos, como saber emque casos ele é solucionável ou não?". Apresentaremos um problema assim aqui. É a versão do ***halting problem***, ou **problema da parada**, para a atual definição de máquinas de Turing.

Como já vimos em seção anterior, tendo uma quantidade enumerável de alfabetos, a quantidade de máquinas de Turing que se podem originar a partir desses alfabetos também é enumerável. Acontece que, devido a isso, podemos afirmar que há linguagens surgidas a partir desses alfabetos que não são Turing-reconhecíveis.

Para enxergarmos isso, mostraremos que a quantidade de tais linguagens é não-enumerável. E com isso, a nossa afirmação ganha ainda um outro contorno. Pois não apenas há linguagens que não sejam Turing-reconhecíveis. Mas há mais linguagens que não são Turing-reconhecíveis do que linguagens Turing-reconhecíveis.

Antes de chegar à devida demonstração disso, passamos pelo resultado abaixo.

Teorema 5.4.1. *O conjunto dos números reais é não-enumerável.*

Demonstração. Suporemos que o conjunto $\mathbb{R}$ é enumerável. Sendo assim, como se trata de um conjunto infinito, é possível construirmos uma bijeção f entre $\mathbb{N}$ e $\mathbb{R}$. E então, todo número real deve ser imagem de algum natural através da função f.

Teríamos, assim, uma sequência de reais sendo formada, como no exemplo abaixo.

$$f(0) = 14,34400000\ldots$$
$$f(1) = -2,79454545\ldots$$
$$f(2) = 103,12165702\ldots$$
$$f(3) = 78,09982453\ldots$$
$$f(4) = 2000,6300000\ldots$$
$$\vdots$$

Só que agora, mostraremos que há pelo menos um número real que não está na sequência. Seja $i \in \mathbb{N}$. Vamos tomar um número real $x \in (0,1)$. Ele será tal que pode ser representado, na base decimal, de modo que o $(i+1)$-ésimo algarismo após a vírgula de x será distinto do $(i+1)$-ésimo algarismo após a vírgula de $f(i)$. Além disso, definimos que x não possui algarismos 0 e nem 9.

Este detalhe final é colocado para que x não seja igual a nenhum número da sequência. Pois x poderia ter apenas a representação decimal distinta, mas guardar o mesmo valor que algum outro real sequenciado.

Afinal, sabemos que, na base 10, podemos escrever certos racionais tanto com infinitos dígitos 0 seguidos a partir de uma posição, e escrever o mesmo número

com infinitos dígitos 9 seguidos a partir de uma posição. Por exemplo, os números $3,599999\ldots$ e $3,600000\ldots$ são iguais. Já com a nossa definição, impedimos a repetição de números de aqui acontecer.

Sendo assim, x é diferente de $f(0)$, pois não tem o seu primeiro algarismo após a vírgula igual ao de $f(0)$. Também x é diferente de $f(1)$, pois não tem o seu segundo algarismo após a vírgula igual ao de $f(1)$. x é diferente de $f(2)$, pois não tem o seu terceiro algarismo após a vírgula igual ao de $f(2)$. E assim segue.

A conclusão é que x é um número real que não está na sequência construída a partir de f. Isso significa que a função f não é sobrejetora, e portanto não é bijetora. Chegamos a uma contradição. Logo, o conjunto $\mathbb{R}$ só pode ser não-enumerável. ■

Poderíamos ter apenas utilizado o teorema acima como um resultado previamente aceito. Mas o próprio método que utilizamos para a sua demonstração será recordado posteriormente. Ele é chamado de **método da diagonal**, pois forma como que uma diagonal quando "ligamos" as casas decimais dos algarismos a partir dos quais definíamos nosso número x.

Trazemos uma exemplificação disso no esquema abaixo representado. Os algarismos relevantes para nossa demonstração, como vemos, foram apenas aqueles colocados após a vírgula.

Figura 52 – Visualização para o método da diagonal

$f(0) =$	$14,$	$\boxed{3}$	4	4	0	0	0	0	0	$\cdots$
$f(1) =$	$-2,$	7	$\boxed{9}$	4	5	4	5	4	5	$\cdots$
$f(2) =$	$103,$	1	2	$\boxed{1}$	6	5	7	0	2	$\cdots$
$f(3) =$	$78,$	0	9	9	$\boxed{8}$	2	4	5	3	$\cdots$
$f(4) =$	$2000,$	6	3	0	0	$\boxed{0}$	0	0	0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Fonte: Autor (2024)

Enfim, provamos a afirmação que era originalmente de nosso interesse na demonstração do teorema abaixo.

Teorema 5.4.2. *Se o conjunto Σ é um alfabeto, então existe uma quantidade não-enumerável de linguagens a partir de Σ.*

Demonstração. Tomemos o caso em que $\Sigma = \{\alpha\}$ é o nosso alfabeto, onde α é um símbolo qualquer. Sabemos que as linguagens existentes a partir deste alfabeto são os subconjuntos de Σ^*.

Tomando qualquer alfabeto com mais de um símbolo, o seu conjunto das partes evidentemente não terá quantidade de elementos menor do que a do conjunto das partes de Σ^*. Portanto, para provarmos o teorema, é suficiente provarmos que $\mathcal{P}(\Sigma^*)$ é não-enumerável.

Ora, podemos construir um função sobrejetora que leve os elementos de $\mathcal{P}(\Sigma^*)$ até o conjunto dos números reais de zero a um.

Bom, sabemos que $\Sigma^* = \{\varepsilon, \alpha, \alpha\alpha, \alpha\alpha\alpha, \alpha\alpha\alpha\alpha, \dots\}$. Sejam então $s_0 = \varepsilon$, $s_1 = \alpha$, $s_2 = \alpha\alpha$, $s_3 = \alpha\alpha\alpha$, e assim por diante. Podemos tomar uma bijeção f que identifica cada linguagem A, tomada a partir do alfabeto Σ, com uma sequência infinita de dígitos 0 e 1.

Seja $i \in \mathbb{N}$. Para a bijeção que apontamos acima, cada sequência dessas será montada de maneira que a sequência correspondente à linguagem A terá o seu $(i+1)$--ésimo dígito igual a 1 se o elemento $s_i \in A$, e terá o seu $(i+1)$-ésimo dígito igual a 0 se o elemento $s_i \notin A$. Denotaremos tal sequência por χ_A.

Então, por exemplo, se, colocando na ordem que apresentamos os elementos de Σ^* anteriormente, tivermos $A = \{\varepsilon, \alpha\alpha, \alpha\alpha\alpha, \alpha\alpha\alpha\alpha\alpha, \dots\}$, então teremos $\chi_A = 1011001\dots$

Vendo isso, cada linguagem A é determinada exatamente pelos elementos que ela traz e pelos que não traz de Σ^*. Daí, cada linguagem vai ter sua sequência correspondente sendo distinta da sequência correspondente de cada uma das demais. E também, cada sequência representa um subconjunto distinto de Σ^*, pois seus dígitos posicionados mostram quais elementos A possui e quais não possui, definindo tal linguagem como diferente das demais.

Por isso, sendo $\mathcal{B}$ o conjunto das sequências infinitas formadas apenas pelos dígitos 0 e 1, temos a bijeção $f : \Sigma^* \to \mathcal{B}$, dada por $f(A) = \chi_A$.

E, sendo $Y \in \mathcal{B}$, diremos que $Y_{(i)}$ é o $(i+1)$-ésimo dígito elemento da sequência Y. Agora construiremos uma função sobrejetora g que leva cada sequência de $\mathcal{B}$ até um número real de zero a um.

Será de modo que tal número terá a parte inteira representada por 0 e, na base binária, terá a sua parte fracionária formada pelo $(i+1)$-ésimo dígito à direita da vírgula igual a $Y_{(i)}$. Isto é, se $Y = Y_{(0)}Y_{(1)}Y_{(2)}Y_{(3)}\dots$, então o número real atribuído a essa sequência será $\left(0, Y_{(0)}Y_{(1)}Y_{(2)}Y_{(3)}\dots\right)_2$.

Como toda sequência dessas é um elemento χ_A, diremos que o número real atribuído a ela será o número r_A. E a nossa função $g : \mathcal{B} \to [0, 1]$ será dada por $g(\chi_A) = r_A$. Vemos que g é sobrejetora porque todo número de zero a um é representado por alguma sequência infinita dos dígitos binários.

Recordando o teorema 3.2.1, os nossos números irracionais serão aqueles cuja sequência é não-periódica. Os números racionais serão aqueles cuja sequência, a partir de algum momento, é periódica, e isso inclui os racionais que normalmente seriam representados com finitos dígitos. Pois eles possuem uma representação equivalente com infinitos dígitos 0 após aquele que seria normalmente o último dígito, ou então eles possuem uma representação com infinitos dígitos 1 seguidos a partir de certa posição. Este último é, por exemplo, o caso do número real 1, cuja representa-

ção aqui seria $(0, 11111\ldots)_2$.

Agora, tomando a função $h = g \circ f$, temos uma função sobrejetora de $\mathcal{P}(\Sigma^*)$ em $[0, 1]$. E como o contradomínio dessa função é não-enumerável, isso significa que o conjunto de nossas linguagens também é não-enumerável. ■

Observação 6. Podemos provar que o conjunto $[0, 1]$ é não-enumerável de maneira similar a como provamos para $\mathbb{R}$. Suporíamos que o conjunto é enumerável e então listaríamos apenas elementos seus na sequência. E aplicaríamos o argumento com o método da diagonal para mostrar a contradição.

Corolário 5.4.3. *Existem linguagens que não são Turing-reconhecíveis.* ■

Mais a seguir, vamos tratar do *halting problem*. Agora, seja Σ um alfabeto capaz de codificar os objetos que vêm na sequência. Começaremos a trabalhar em cima de tal problema através de uma linguagem que remete a um outro problema. Trata-se da linguagem

$$\mathsf{A_{MT}} = \{\langle \mathcal{M}, w\rangle \in \Sigma^*;\ \mathcal{M} \text{ é uma máquina de Turing e } \mathcal{M} \text{ aceita } w\}.$$

Antes disso, introduziremos a máquina de Turing $\mathcal{U}$, inspirada na antes apresentada máquina de Turing universal. A máquina $\mathcal{U}$ já visa simular máquinas de Turing quaisquer. A seguir, sua descrição.

$\mathcal{U}$ = " Sobre a entrada $\langle \mathcal{M}, w\rangle$, onde $\mathcal{M}$ é uma máquina de Turing e w uma cadeia:

1. Simule a máquina $\mathcal{M}$ sobre a entrada w.
2. Se a máquina $\mathcal{M}$, em algum momento, entra em seu estado de aceitação, então aceite. E se $\mathcal{M}$, em algum momento, entra em seu estado de rejeição, então rejeite."

Assim como a máquina $\mathcal{U}^{\mathcal{T}}$ buscava quais eram os estados em que uma máquina $\mathcal{M}^{\mathcal{T}}$ simulada entraria com cada configuração atual ao comparar tal configuração com as instruções que $\mathcal{M}^{\mathcal{T}}$ possuía, podemos ter a máquina $\mathcal{U}$ fazendo trabalho análogo com $\mathcal{M}$.

Observação 7. Lembrando que as máquinas chamadas de $\mathcal{M}^{\mathcal{T}}$ são aquelas que pertencem à classe das máquinas tais como o próprio Alan Turing definiu, com o objetivo de imprimir os F-símbolos 0 e 1, com o delimitador ə no início da fita, etc.

Consideremos, como uma das possibilidades para esta máquina, aquela onde ela funciona a partir dos termos do alfabeto de codificação que já estamos vendo

desde $\mathcal{U}^{\mathcal{T}}$, com alguns acréscimos. Após a codificação da máquina $\mathcal{M}$, vem a codificação da cadeia w, em termos de D e C. Teria apenas um D se fosse a cadeia vazia.

A máquina $\mathcal{U}$ verificará se a entrada que recebeu está realmente na forma de representar uma máquina de Turing seguida de uma entrada para seu respectivo alfabeto. Depois, quando chegar o momento em que, tal como $\mathcal{U}^{\mathcal{T}}$, a máquina $\mathcal{U}$ levar para o fim da fita a primeira configuração completa da simulação, em vez de simplesmente fazê-lo com o estado inicial e um espaço vazio, ela o fará com o estado inicial precedendo toda a cadeia de entrada w. Depois disso os procedimentos são como na máquina $\mathcal{U}^{\mathcal{T}}$.

Só que, no caso da máquina $\mathcal{U}^{\mathcal{T}}$, ela simulava máquinas $\mathcal{M}^{\mathcal{T}}$, e estas já tinham um delimitador para indicar o início da fita, que era o símbolo ə. Os estados de uma máquina $\mathcal{M}^{\mathcal{T}}$ já eram tais que faziam a cabeça voltar para a direita quando escaneavam tal símbolo. E, desse modo, quando $\mathcal{U}^{\mathcal{T}}$ simulava as máquinas, não tinha o risco de ir para trás do que seria o início da fita de $\mathcal{M}^{\mathcal{T}}$.

Para a máquina $\mathcal{U}$ simular máquinas de Turing $\mathcal{M}$ quaisquer e não ir para trás do que seria o início de suas fitas, é necessário recorrer a algum artifício, visto que não há obrigação de as máquinas simuladas terem algum delimitador do seu início. Mas é simples resolver isso.

Basta ajustar $\mathcal{U}$ para que, quando ela estiver imprimindo a primeira das configurações completas de $\mathcal{M}$ ao fim da fita, o primeiro símbolo dela, antes dos outros, seja um I. Este símbolo deverá ser reimpresso em cada nova configuração completa ao fim da fita do mesmo jeito que os outros D e C não referentes ao símbolo atualmente impresso em $\mathcal{M}$ são reimpressos em $\mathcal{U}$.

Até então não usado neste contexto, o I indicará o início da fita da máquina $\mathcal{M}$. Se, na próxima configuração completa, os símbolos referentes ao estado de $\mathcal{M}$ precederem I, estaríamos no caso em que a cabeça da máquina simulada já estava no início de sua fita e tentou andar para a esquerda.

Se isso acontecer, a máquina $\mathcal{U}$ deve apagar esta configuração completa ao final e substitui-la pela versão em que os símbolos referentes ao estado atual, em vez de estarem imediatamente antes de I, estejam imediatamente após I, e depois venham os símbolos representando a cadeia da fita de $\mathcal{M}$. Isso para garantir que a simulação vai trazer a cabeça de $\mathcal{M}$ ainda dentro dos limites da fita simulada.

Ainda, há a questão dos estados de aceitação e de rejeição. Conseguiríamos, por exemplo, representar o estado de aceitação de $\mathcal{M}$, na codificação com os símbolos DE, o estado de rejeição com os símbolos DEE, e os demais estados de $\mathcal{M}$ com um símbolo D seguido de símbolos A. Ajustando os estados de $\mathcal{U}$ para reconhecer tais estados de $\mathcal{M}$ quando comparar a configuração da fita virtual com as instruções de $\mathcal{M}$, podemos determinar que $\mathcal{U}$ entre em seu estado de aceitação quando verificar

que o próximo estado de $\mathcal{M}$ seria o de aceitação; e que $\mathcal{U}$ entre em seu estado de rejeição quando verificar que o próximo estado de $\mathcal{M}$ seria o de rejeição.

E também há a questão do alfabeto de entrada. Nas máquinas $\mathcal{M}^{\mathcal{T}}$, o alfabeto de entrada era vazio, então os únicos símbolos da máquina eram aqueles que pertenciam ao alfabeto da fita. Mas em uma máquina qualquer, temos que estar preparados para codificar símbolos que estejam apenas no alfabeto da fita, pois sem distinguir entre os dois alfabetos, a mesma máquina pode ter uma linguagem ou outra.

Dito isto, uma das maneiras de aplicar isso é a seguinte. Continuamos representando $\llcorner\!\lrcorner$, que pertence ao alfabeto da fita, por D e os símbolos seguintes por DC, DCC, DCCC,... Bom, o alfabeto da fita tem m símbolos, $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$, onde $\alpha_0 = \llcorner\!\lrcorner$. E o alfabeto de entrada tem $m' < m$ símbolos, $\alpha_1, \alpha_2, \ldots, \alpha_{m'}$.

Convencionaremos que os símbolos representados desde o DC até o representado por $D\overbrace{\text{CCC}\ldots\text{CC}}^{m' \text{ vezes}}$, são todos os símbolos do alfabeto de entrada. E os símbolos seguintes representados que houver são símbolos exclusivos do alfabeto da fita, junto do $\llcorner\!\lrcorner$.

Após as instruções todas das configurações que traz a descrição de $\mathcal{M}$, iremos colocar uma quantidade de m' símbolos B, espaçados entre si por um quadrado preenchido com $\widehat{\llcorner\!\lrcorner}$. Isto servirá para indicar que apenas os símbolos que forem representados com até m' caracteres C podem fazer parte de uma cadeia de entrada da máquina.

A máquina universal, quando traz uma máquina $\mathcal{M}$ e uma entrada w, pode fazer tal verificação quanto ao alfabeto de entrada comparando a quantidade de C's dos símbolos de w com a quantidade de B's. Isto funciona mesmo se tal alfabeto for vazio. A máquina $\mathcal{U}$ procurará algum B do início ao fim da descrição de $\mathcal{M}$ e não encontrará.

Nas máquinas de Turing em geral já não há a obrigação de importar-se com F-quadrados e E-quadrados. E nem há um conjunto universal e finito do que seriam F-símbolos. Mas, neste âmbito, o que costuma mais importar é se haverá aceitação ou rejeição. Sendo o papel de $\mathcal{U}$ captar quando acontecem, é desnecessário se preocupar, para tal máquina, com o papel que a máquina $\mathcal{U}^{\mathcal{T}}$ tinha em imprimir, depois, os símbolos do alfabeto em si mesmos. E muito menos se haveria impressão em cima de quadrados já preenchidos anteriormente.

Agora, retornando à linguagem antes definida, vejamos primeiro que ela é Turing-reconhecível.

Proposição 5.4.4. *A linguagem A_{MT} é Turing-reconhecível.*

Demonstração. Afirmamos que a máquina de Turing $\mathcal{U}$ reconhece a linguagem em questão.

Ora, a máquina $\mathcal{U}$ aceita uma entrada $\langle \mathcal{M}, w \rangle$ exatamente quando $\mathcal{M}$ aceita w;

ela rejeita $\langle \mathcal{M}, w \rangle$ quando $\mathcal{M}$ rejeita w; e ela entra em loop a partir de $\langle \mathcal{M}, w \rangle$ quando $\mathcal{M}$ entra em loop a partir de w.

Sendo assim, ela reconhece A_{MT}. ■

É de nosso interesse aqui notarmos que, a máquina $\mathcal{U}$ reproduz o aspecto do final do trabalho da máquina $\mathcal{M}$ para a cadeia w. Isto é, se o trabalho acaba em rejeição, em aceitação, ou se ele simplesmente não acaba. E, justamente por poder não acabar, a máquina $\mathcal{U}$ não é um decisor.

E aqui reside o chamado ***accepting problem***, ou, problema da aceitação. Com $\mathcal{U}$, só saberemos se a máquina $\mathcal{M}$ pararia seu trabalho sobre uma cadeia se em algum momento $\mathcal{U}$ também parar. Não estamos com um mecanismo que consiga rejeitar a máquina $\mathcal{M}$ e entrada w quando, juntas, acabariam por entrar em loop.

Encontrar um mecanismo para isso seria, em outras palavras, encontrar uma máquina de Turing que decidisse $\langle \mathcal{M}, w \rangle$, aceitando tal entrada quando $\mathcal{M}$ aceitasse w, e a rejeitando nos outros casos. Ou seja, uma máquina de Turing que decidisse a linguagem A_{MT}.

Todavia, a questão é que tal linguagem não é decidível, ou, se quisermos dizer, **indecidível**. Enunciamos tal teorema, o qual provamos daqui a pouco.

Teorema 5.4.5. *A linguagem A_{MT} é indecidível.*

Vamos demonstrá-lo daqui a pouco, porque primeiro queremos dar um exemplo de algo até então deixado de lado. Chegamos já à conclusão de que existem linguagens que não são Turing-reconhecíveis, ou, se quisermos, **Turing-irreconhecíveis**. Mas não vimos nenhuma delas.

E, tomando a afirmação de que a linguagem A_{MT} é indecidível, podemos chegar a uma linguagem Turing-irreconhecível. E esta linguagem se trata de nada mais nada menos que $\overline{A_{MT}}$. A seguir.

Teorema 5.4.6. *A linguagem $\overline{A_{MT}}$ é Turing-irreconhecível.*

Demonstração. Pelo teorema 5.3.3, se uma linguagem é Turing-reconhecível e co--Turing-reconhecível, então ela é decidível. Pela proposição 5.4.4, a linguagem A_{MT} é Turing-reconhecível. Se ela fosse co-Turing-reconhecível, então ela teria também que ser decidível. Mas isto é falso. Portanto, A_{MT} não é co-Turing-reconhecível. Ou seja, $\overline{A_{MT}}$ é Turing-irreconhecível. ■

Agora que temos o nosso exemplo de linguagem que nenhuma máquina de Turing reconhece, podemos prosseguir com a demonstração do teorema 5.4.5.

Nessa demonstração, utilizaremos, em dado momento, o modo de trabalhar de uma máquina de Turing dentro de outra máquina de Turing. Dizemos que este mecanismo interior a uma máquina $\mathcal{M}_1$ que age como uma máquina $\mathcal{M}_2$ é uma **subrotina**.

Mais especificamente, a máquina $\mathcal{M}_2$ será subrotina da máquina $\mathcal{M}_1$. A partir de certo momento, $\mathcal{M}_1$ vai agir do jeito que $\mathcal{M}_2$ agiria, para cumprir uma determinada tarefa. Este conceito aparecerá outras vezes.

Demonstração do teorema 5.4.5. Suponhamos que $\mathsf{A_{MT}}$ seja decidível. Então existe uma máquina $\mathcal{H}$ para a qual a entrada $\langle \mathcal{M}, w \rangle$ é aceita se $\mathcal{M}$ aceita w, e é rejeitada se $\mathcal{M}$ não aceita w, seja porque $\mathcal{M}$ rejeita w ou porque ela entra em loop a partir de tal entrada.

Agora, tomaremos uma máquina de Turing $\mathcal{D}$ da qual podemos dizer que $\mathcal{H}$ é uma subrotina. A máquina $\mathcal{D}$ trabalha em cima das codificações de outras máquinas de Turing $\mathcal{M}$, isto é, de entradas $\langle \mathcal{M} \rangle$.

$\mathcal{D}$ = **"** Sobre a entrada $\langle \mathcal{M} \rangle$, onde $\mathcal{M}$ é uma máquina de Turing capaz de receber como entradas as codificações de outras máquinas de Turing:

1. Emule $\mathcal{H}$ sobre a entrada $\langle \mathcal{M}, \langle \mathcal{M} \rangle \rangle$.
2. Em vez dos estados finais que $\mathcal{H}$ daria sobre esta entrada, tenha que, se $\mathcal{H}$ originalmente rejeitaria, então aceite, e se $\mathcal{H}$ originalmente aceitaria, então rejeite.**"**

Para que $\mathcal{D}$ reconheça se a máquina de Turing codificada em sua entrada realmente consegue receber as entradas mencionadas, basta verificar se o alfabeto de entrada dela possui a quantidade de elementos maior ou igual ao tamanho do alfabeto que estejamos adotando para a codificação. Afinal, diferentes alfabetos de mesmo tamanho serão codificados de mesma maneira, então podemos considerar que aqueles que atendem o requisito acima cumprem o mesmo papel de codificar.

Expliquemos o funcionamento da máquina $\mathcal{D}$, sobre uma entrada $\langle \mathcal{M} \rangle$. Ela age como a máquina $\mathcal{H}$, que verifica se a máquina $\mathcal{M}$ aceita a cadeia de sua própria codificação $\langle \mathcal{M} \rangle$, mas faz o contrário ao final, rejeitando as entradas que $\mathcal{H}$ aceitaria, e aceitando todas as outras.

Assim, para uma máquina de Turing $\mathcal{M}$, se $\mathcal{H}$ aceitasse a entrada $\langle \mathcal{M}, \langle \mathcal{M} \rangle \rangle$, isso significa que a máquina $\mathcal{D}$ vai rejeitar a entrada $\langle \mathcal{M} \rangle$; e se $\mathcal{H}$ rejeitasse a entrada $\langle \mathcal{M}, \langle \mathcal{M} \rangle \rangle$, isso significa que a máquina $\mathcal{D}$ vai aceitar a entrada $\langle \mathcal{M} \rangle$.

E, pela definição de $\mathcal{H}$, podemos equivalentemente falar que, se $\mathcal{M}$ aceita a entrada $\langle \mathcal{M} \rangle$, isso significa que a máquina $\mathcal{D}$ vai rejeitar a entrada $\langle \mathcal{M} \rangle$; e se $\mathcal{M}$ não aceita a entrada $\langle \mathcal{M} \rangle$, isso significa que a máquina $\mathcal{D}$ vai aceitar a entrada $\langle \mathcal{M} \rangle$.

Entretanto, se a máquina de Turing $\mathcal{D}$ faz isso ao ter como entrada a codificação de uma máquina de Turing qualquer, acontecerá algo de errado quando a sua entrada for a codificação da própria $\mathcal{D}$. Porque, pelo que acabamos de afirmar, decorre que, se $\mathcal{D}$ aceita a entrada $\langle \mathcal{D} \rangle$, isso significa que a máquina $\mathcal{D}$ vai rejeitar a entrada $\langle \mathcal{D} \rangle$;

e se $\mathcal{D}$ não aceita a entrada $\langle\mathcal{D}\rangle$, isso significa que a máquina $\mathcal{D}$ vai aceitar a entrada $\langle\mathcal{D}\rangle$.

Como isto é contraditório, segue que a linguagem A_{MT} não é decidível. ■

E ainda, podemos dizer que o método da diagonal está implicitamente presente na demonstração acima. Pois, presumindo que fosse possível termos as máquinas $\mathcal{H}$ e $\mathcal{D}$, teríamos uma tabela que representasse os estados finais das máquinas de Turing a partir das entradas correspondentes às codificações de cada máquina de Turing.

Lembremos mais uma vez que, como vimos no teorema 3.6.1, tomando uma quantidade enumerável de alfabetos da fita para as máquinas de Turing, teremos uma quantidade também enumerável de máquinas de Turing advindas destes. Assim, generalizemos todos os alfabetos de $m \in \mathbb{N} - \{0\}$ símbolos como se fossem o mesmo alfabeto da fita. Afinal, o que importa aqui não é a natureza do símbolo, e todas as máquinas de Turing que surgirem de um único desses alfabetos já guardariam todas as possibilidades de ação, apenas rearranjando os símbolos segundo suas posições.

Partindo disso, poderíamos enumerar todas as máquinas de Turing existentes. Aqui nos limitamos a enumerar as máquinas cujo alfabeto de entrada é suficientemente grande para abarcar os símbolos do alfabeto de codificação de máquinas de Turing que estivermos utilizando. Tomando-as então, as chamaremos de $\mathcal{M}_0$, $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, assim seguindo. E, primeiramente, trazemos um exemplo do que seria a tabela que aponta quando uma máquina de Turing específica aceita, rejeita ou entra em loop a partir da entrada de cada codificação das máquinas de Turing.

Figura 53 – Exemplificação do resultado das computações das máquinas sobre as codificações

	$\langle\mathcal{M}_0\rangle$	$\langle\mathcal{M}_1\rangle$	$\langle\mathcal{M}_2\rangle$	$\langle\mathcal{M}_3\rangle$	$\cdots$
$\mathcal{M}_0$	aceita	loop	aceita	rejeita	$\cdots$
$\mathcal{M}_1$	aceita	rejeita	aceita	loop	$\cdots$
$\mathcal{M}_2$	loop	rejeita	aceita	aceita	$\cdots$
$\mathcal{M}_3$	aceita	loop	rejeita	rejeita	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Fonte: Autor (2024)

A linha representa a máquina que está computando, e a coluna representa uma entrada de tal máquina.

Agora, tomando o mesmo exemplo, podemos construir uma segunda tabela, a qual aponta os estados finais da máquina $\mathcal{H}$.

Em tal tabela, a linha relativa a uma máquina de Turing $\mathcal{M}_i$ e a coluna relativa a uma cadeia $\langle\mathcal{M}_j\rangle$ dizem respeito, juntas, a se a máquina $\mathcal{H}$ vai aceitar ou rejeitar a entrada $\langle\mathcal{M}_i, \langle\mathcal{M}_j\rangle\rangle$. Ou seja, respectivamente, elas apontam se $\mathcal{M}_i$ aceita $\langle\mathcal{M}_j\rangle$ ou se não a aceita.

Figura 54 – Exemplificação do resultado da computação de $\mathcal{H}$ sobre as máquinas com as entradas

"$\mathcal{H}$"	$\langle\mathcal{M}_0\rangle$	$\langle\mathcal{M}_1\rangle$	$\langle\mathcal{M}_2\rangle$	$\langle\mathcal{M}_3\rangle$	$\cdots$
$\mathcal{M}_0$	aceita	rejeita	aceita	rejeita	$\cdots$
$\mathcal{M}_1$	aceita	rejeita	aceita	rejeita	$\cdots$
$\mathcal{M}_2$	rejeita	rejeita	aceita	aceita	$\cdots$
$\mathcal{M}_3$	aceita	rejeita	rejeita	rejeita	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Fonte: Autor (2024)

E, por último, apresentamos a mesma tabela, mas levando em conta que a máquina $\mathcal{D}$ também está entre as máquinas de Turing enumeradas. A máquina $\mathcal{D}$ vai rejeitar ou aceitar as entradas dependendo exclusivamente do que acontecer na diagonal principal de nossa tabela. Mais precisamente, na linha de $\mathcal{D}$ com a coluna de $\langle\mathcal{M}_j\rangle$, teremos o oposto do que temos na linha de $\mathcal{M}_j$ com a coluna de $\langle\mathcal{M}_j\rangle$.

O preenchimento das primeiras células na coluna de $\langle\mathcal{D}\rangle$ é hipotético e não guarda relação com as demais computações presentes na tabela.

Figura 55 – Exemplificação do resultado da computação de $\mathcal{H}$ sobre as máquinas com as entradas explicitada até a máquina $\mathcal{D}$

"$\mathcal{H}$"	$\langle\mathcal{M}_0\rangle$	$\langle\mathcal{M}_1\rangle$	$\langle\mathcal{M}_2\rangle$	$\langle\mathcal{M}_3\rangle$	$\cdots$	$\langle\mathcal{D}\rangle$	$\cdots$
$\mathcal{M}_0$	aceita	rejeita	aceita	rejeita	$\cdots$	aceita	$\cdots$
$\mathcal{M}_1$	aceita	rejeita	aceita	rejeita	$\cdots$	aceita	$\cdots$
$\mathcal{M}_2$	rejeita	rejeita	aceita	aceita	$\cdots$	rejeita	$\cdots$
$\mathcal{M}_3$	aceita	rejeita	rejeita	rejeita	$\cdots$	rejeita	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$
$\mathcal{D}$	rejeita	aceita	rejeita	aceita	$\cdots$	**?**	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$

Fonte: Autor (2024)

E o grande revés aqui é que, então, na linha de $\mathcal{D}$ com a coluna de $\langle\mathcal{D}\rangle$ teríamos o oposto do que temos na linha de $\mathcal{D}$ com a coluna de $\langle\mathcal{D}\rangle$. Mas isso não faz sentido, visto que temos apenas as opções de aceitar e rejeitar. Neste ponto da diagonal não poderíamos aceitar e rejeitar a mesma coisa ao mesmo tempo.

Pois agora, que estamos em posse do resultado de que o *accepting problem* é indecidível, podemos chegar à mesma conclusão quanto ao *halting problem*. Trata-se de conseguir decidir quando alguma máquina de Turing, sobre uma determinada entrada, para ou não. Parar significa quando entra no estado de aceitação ou no de rejeição.

Para tal, definimos a linguagem

$$\text{PARA}_{\text{MT}} = \{\langle \mathcal{M}, w\rangle \in \Sigma^*;\ \mathcal{M} \text{ é uma máquina}$$

$$\text{de Turing e } \mathcal{M} \text{ para sobre a entrada } w\}.$$

Teorema 5.4.7. *A linguagem PARA$_{MT}$ é indecidível.*

Demonstração. Mostraremos que, se a linguagem em questão fosse decidível, então a linguagem A_{MT} também o seria, o que contradiz o teorema 5.4.5. Para mostrá-lo, construiremos a máquina hipotética que decide esta última.

Supondo, então, ser PARA_{MT} decidível, isso significa que existe uma máquina $\mathcal{R}$ que decide PARA_{MT}. Ou seja, $\mathcal{R}$ aceita somente as entradas $\langle \mathcal{M}, w\rangle$ onde a máquina $\mathcal{M}$ aceita ou rejeita a entrada w, enquanto rejeita as demais entradas.

Só que, se isso é verdade, poderíamos construir uma máquina $\mathcal{S}$ como descrita abaixo.

$\mathcal{S}$ = **“** Sobre a entrada $\langle \mathcal{M}, w\rangle$, onde $\mathcal{M}$ é uma máquina de Turing e w é uma cadeia:

1. Emule a máquina $\mathcal{R}$ sobre a entrada $\langle \mathcal{M}, w\rangle$.
2. Se $\mathcal{R}$ rejeita, então rejeite.
3. Se $\mathcal{R}$ aceita, então simule $\mathcal{M}$ sobre a entrada w até o final.
4. Se $\mathcal{M}$ aceitou a entrada w, então aceite, e se $\mathcal{M}$ a rejeitou, então rejeite.**”**

Acontece que esta máquina decide A_{MT}. Pois as entradas $\langle \mathcal{M}, w\rangle$ rejeitadas por $\mathcal{R}$, e consequentemente por $\mathcal{S}$, são aquelas em que $\mathcal{M}$ entra em loop sobre w. As entradas aceitas por $\mathcal{R}$ e rejeitadas por $\mathcal{S}$ são aquelas em que $\mathcal{M}$ rejeita a entrada w. E as aceitas por $\mathcal{S}$ são todas as entradas em que $\mathcal{M}$ aceita a entrada w.

Isso significa que $\mathcal{S}$ aceita qualquer entrada $\langle \mathcal{M}, w\rangle$ em que $\mathcal{M}$ aceita w, e rejeita todas as outras entradas. Isto é, $\mathcal{S}$ decide a linguagem A_{MT}. Como isto não é verdadeiro, segue que PARA_{MT} é indecidível. ■

Com a demonstração da indecidibilidade, tanto do *halting problem* como do *accepting problem*, a partir da hipótese de que a tese de Church-Turing esteja correta, chegamos a enxergar já uma barreira diante de problemas como estes. Pois, sendo tais problemas indecidíveis, então não há o que possamos fazer para resolvê-los universalmente. Podemos conseguir dar conta de casos particulares, mas, tal como não há uma máquina de Turing que decida tal linguagem, não há um algoritmo que vá dar conta de todo o problema.

Ao mesmo tempo, entendemos que, se tentássemos insistir em encontrar uma solução universal para isto, estaríamos perdendo nosso tempo, investindo inutilmente nossos esforços. Tendo o conhecimento dessa limitação, podemos utilizar nossas energias e recursos em outros problemas. Isto, é claro na hipótese de tal tese ser verdadeira, algo de que, a princípio, não possuímos uma certeza.

6 OU MENOS ESTADOS OU MENOS SÍMBOLOS

Vejamos, agora, algumas alternativas para a computação de certas máquinas. Imaginemos uma máquina cujos alfabetos (da fita e de entrada) têm uma maior quantidade de elementos, como também um conjunto de estados com mais elementos. Podemos escolher fazer um trabalho análogo ao dela, em outra máquina, diminuindo ou os seus estados ou os seus símbolos.

6.1 UMA MÁQUINA E DOIS ESTADOS ABRANGENTES

Na seção anterior introduzimos a máquina universal $\mathcal{U}$. Definida de um modo mais genérico, ela conta com um alfabeto da fita de vários símbolos e com vários estados. O que mostraremos nesta seção é que qualquer máquina de Turing, inclusive ela, pode ser replicada de modo a ter somente dois estados que operam tudo, fora a aceitação e a rejeição. Isto é guiado pela publicação de Shannon (1956).

Para cada máquina de Turing, montaremos uma máquina de Turing correspondente, com seus símbolos sendo multiplicados de modo a permitir que apenas dois estados deem conta do que, em geral, a máquina original faria. Sendo $\mathcal{M}$ a máquina original, chamaremos a nova máquina de $\mathcal{M}_{[2]}$.

Haverá símbolos em $\mathcal{M}_{[2]}$ que representam não simplesmente os símbolos de $\mathcal{M}$, mas a configuração em que $\mathcal{M}$ estaria naquele estágio. E também haverá os símbolos que fazem a transição de um símbolo/configuração para outro entre quadrados adjacentes.

Os nossos dois estados de $\mathcal{M}_{[2]}$ serão chamados de $\mathfrak{a}$ e $\mathfrak{b}$, responsáveis, em alguns casos, por levar a cabeça respectivamente para a direita e para a esquerda. Tomando a máquina $\mathcal{M}$, seja $\Gamma = \{\alpha_0, \alpha_1, \ldots, \alpha_{m-1}\}$ o seu alfabeto da fita, com $\alpha_0 = \text{␣}$, e $Q = \{\mathfrak{q}_0, \mathfrak{q}_1\ \ldots, \mathfrak{q}_{n-1}\}$ o seu conjunto de estados. Seja, também, $m' < m$ o número de elementos do alfabeto de entrada de $\mathcal{M}$, sendo ele $\Sigma = \{\alpha_1, \alpha_2, \ldots, \alpha_{m'}\}$. E δ a sua função de transição. Eis como montaremos a nova máquina.

Aos $(m-1)$ símbolos de $\mathcal{M}$ que não α_0, teremos $(m-1)$ símbolos correspondentes, $\beta_1, \beta_2, \ldots, \beta_{m-1}$ em $\mathcal{M}_{[2]}$. E teremos os dois símbolos β_0 e β_m, ambos para se referir ao que seria α_0, mas com diferentes papéis, sendo β_0 o próprio ␣.

Na máquina $\mathcal{M}_{[2]}$ teremos ainda outros $4mn$ símbolos, que serão denotados por $\beta_{h(i),j,x,y}$, sendo os índices $i \in \{0, 1, \ldots, m-1\}$, $j \in \{0, 1, \ldots, n-1\}$, $x \in \{+, -\}$,

$y \in \{\mathsf{R}, \mathsf{L}\}$, e onde

$$h(i) = \begin{cases} i, & \text{se } i \neq 0, \\ m, & \text{se } i = 0 \end{cases}.$$

O valor de i indica o símbolo original, o valor de j indica o estado original, o valor de x indica se o quadrado em questão está transmitindo $(+)$ ou recebendo $(-)$ informação, e o valor de y indica a direção para a qual a cabeça da máquina irá a partir daquele quadrado.

Com tais símbolos faremos a transição entre o que seria uma configuração na máquina original para outra configuração, e chamaremos este trabalho de transição da máquina $\mathcal{M}_{[2]}$ de **operação de salto**.

No entanto, havemos de tomar símbolos que sirvam para o início da fita, e logo mais o justificaremos. Dados os m' símbolos do alfabeto de entrada original, teremos os símbolos correspondentes $\gamma_1, \gamma_2, \ldots, \gamma_{m'}$, e também $\pi_1, \pi_2, \ldots, \pi_{m'}$ e $\rho_1, \rho_2, \ldots, \rho_{m'}$. Haverá um símbolo γ_0, com um papel único dentre os demais. Também para o início da fita, teremos os símbolos $\lambda_1, \lambda_2, \ldots, \lambda_m$. Ainda, os símbolos da forma $\lambda_{h(i),j,x,y}$.

Seja, assim, $\Gamma_{[2]}$ o conjunto de todos estes novos símbolos que não estavam em $\mathcal{M}$, mais o símbolo $\beta_0 = \text{␣}$. Seja $\Sigma_{[2]} = \{\beta_1, \beta_2, \ldots, \beta_{m'}, \gamma_1, \gamma_2, \ldots, \gamma_{m'}\}$. E seja $Q_{[2]} = \{\mathfrak{a}, \mathfrak{b}, \mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}}\}$.

Com isso, seja a função $\delta_{[2]} : \Gamma_{[2]} \times Q_{[2]} \to \Gamma_{[2]} \times Q_{[2]} \times \{\mathsf{R}, \mathsf{L}\}$ como mais a seguir definiremos. Esta será a nossa função de transição. E assim consideramos ela, desta vez, pois nos parece mais conveniente usar primeiro o símbolo e em segundo o estado como coordenadas. Isso porque conversa melhor com a notação de $\beta_{h(i),j,x,y}$. Esta que, por sua vez, parece mais confortável, pelo fato de deixar o índice dos símbolos imediatamente após o β, familiar ao que já vínhamos fazendo com os símbolos em geral. Teremos, então, $\mathcal{M}_{[2]} = \big(Q_{[2]}, \Sigma_{[2]}, \Gamma_{[2]}, \delta_{[2]}, \mathfrak{a}, \mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}}\big)$.

Ao todo, teremos $(m+1)+4mn+m'+m'+m'+1+m+4mn = (8n+2)m+3m'+2$ símbolos na máquina $\mathcal{M}_{[2]}$. Designando este número como $M_{[2]}$, temos então que $(8n+2)m < M_{[2]} < (8n+5)m$.

Cabe avisarmos que toda essa construção está sendo feita para quando o estado inicial de $\mathcal{M}$ não era um estado de parada. Pois se ele fosse de rejeição ou aceitação, a máquina simplesmente rejeitaria ou aceitaria tudo. Não precisaríamos de um alfabeto diferente, e poderíamos ficar apenas com os estados de parada.

Quanto à função de transição descrita abaixo, os dois primeiros itens tratam do início de uma operação de salto. Tais transições mostram como um símbolo $\beta_{h(i)}$ será sucedido por outro símbolo que, para a direita ou para a esquerda, representa o começo da recepção das informações para que se imite a próxima configuração da máquina $\mathcal{M}$.

O terceiro item trata da transição de um símbolo transmissor, que não no final da

transmissão, diante de $\mathfrak{a}$ e $\mathfrak{b}$. Ele será substituído pelo próximo símbolo de transmissão até se chegue ao índice 0 quanto ao estado, e para o símbolo receptor ao seu lado, lega o estado $\mathfrak{b}$.

O quarto item trata de um símbolo transmissor também, mas ao fim da transmissão. Por causa disso, já é substituído por um símbolo $\beta_{h(i)}$ e lega o estado $\mathfrak{a}$ para o símbolo receptor ao lado.

O quinto item mostra como um símbolo receptor dá seguimento à operação de salto atual quando encontra o estado $\mathfrak{b}$. E os itens do sexto ao nono mostram, dependentes da transição na máquina original, como diante do estado $\mathfrak{a}$, o símbolo receptor termina tal operação de salto. Ele será substituído por um transmissor a partir do que seria a próxima configuração de $\mathcal{M}$. Isto se o próximo estado em $\mathcal{M}$ não era de parada. Se o era, a máquina $\mathcal{M}_{[2]}$ será conduzida para a respectiva parada.

Os demais itens terão seu papel explicado ao longo dos exemplos. Eles estão aí, em geral, devido à questão do símbolo com que a cadeia de entrada inicia, para que possa haver uma devida reprodução da atividade de $\mathcal{M}$ a partir do presente alfabeto de entrada.

Sejam $\mathfrak{q}_{j_1}$ e $\mathfrak{q}_{j_2}$, respectivamente, os estados de aceitação e de rejeição de $\mathcal{M}$. Sejam também $i' \in \{1, 2, \ldots, m'\}$, $j' \in \{0, 1, \ldots, n-2\}$, $\mathfrak{q} \in \{\mathfrak{a}, \mathfrak{b}\}$, e $J \in \{0, 1, \ldots, n-1\} - \{j_1, j_2\}$. Se a função de transição de $\mathcal{M}$ era δ, e $0 \notin \{j_1, j_2\}$, a nossa função de transição será $\delta_{[2]}$. Abaixo a descrevemos.

1. $\delta_{[2]}\left(\beta_{h(i)}, \mathfrak{a}\right) = \left(\beta_{h(i),0,-,\mathsf{R}}, \mathfrak{a}, \mathsf{R}\right)$.
2. $\delta_{[2]}\left(\beta_{h(i)}, \mathfrak{b}\right) = \left(\beta_{h(i),0,-,\mathsf{L}}, \mathfrak{a}, \mathsf{L}\right)$.
3. $\delta_{[2]}\left(\beta_{h(i),j'+1,+,y}, \mathfrak{q}\right) = \left(\beta_{h(i),j',+,y}, \mathfrak{b}, y\right)$.
4. $\delta_{[2]}\left(\beta_{h(i),0,+,y}, \mathfrak{q}\right) = \left(\beta_{h(i)}, \mathfrak{a}, y\right)$.
5. $\delta_{[2]}\left(\beta_{h(i),j',-,y}, \mathfrak{b}\right) = \left(\beta_{h(i),j'+1,-,y}, \mathfrak{a}, y\right)$.
6. $\delta_{[2]}\left(\beta_{h(i),j,-,y}, \mathfrak{a}\right) = \left(\beta_{h(r),J,+,\mathsf{R}}, \mathfrak{b}, \mathsf{R}\right)$, se $\delta\left(\alpha_i, \mathfrak{q}_j\right) = \left(\alpha_r, \mathfrak{q}_J, \mathsf{R}\right)$.
7. $\delta_{[2]}\left(\beta_{h(i),j,-,y}, \mathfrak{a}\right) = \left(\beta_{h(r),J,+,\mathsf{L}}, \mathfrak{a}, \mathsf{L}\right)$, se $\delta\left(\alpha_i, \mathfrak{q}_j\right) = \left(\alpha_r, \mathfrak{q}_J, \mathsf{L}\right)$.
8. $\delta_{[2]}\left(\beta_{h(i),j,-,y}, \mathfrak{a}\right) = \left(\beta_{h(r)}, \mathfrak{q}_{\mathsf{aceita}}, \mathsf{Y}\right)$, se $\delta\left(\alpha_i, \mathfrak{q}_j\right) = \left(\alpha_r, \mathfrak{q}_{j_1}, \mathsf{Y}\right)$.
9. $\delta_{[2]}\left(\beta_{h(i),j,-,y}, \mathfrak{a}\right) = \left(\beta_{h(r)}, \mathfrak{q}_{\mathsf{rejeita}}, \mathsf{Y}\right)$, se $\delta\left(\alpha_i, \mathfrak{q}_j\right) = \left(\alpha_r, \mathfrak{q}_{j_2}, \mathsf{Y}\right)$.
10. $\delta_{[2]}\left(\beta_0, \mathfrak{a}\right) = \left(\gamma_0, \mathfrak{b}, \mathsf{L}\right)$.
11. $\delta_{[2]}\left(\gamma_0, \mathfrak{a}\right) = \left(\beta_0, \mathfrak{q}_{\mathsf{rejeita}}, \mathsf{L}\right)$.
12. $\delta_{[2]}\left(\gamma_0, \mathfrak{b}\right) = \left(\lambda_{h(r),J,+,\mathsf{Y}}, \mathfrak{b}, \mathsf{Y}\right)$, se $\delta\left(\alpha_0, \mathfrak{q}_0\right) = \left(\alpha_r, \mathfrak{q}_J, \mathsf{Y}\right)$.

13. $\delta_{[2]}\left(\gamma_0, \mathfrak{b}\right) = \left(\lambda_{h(r)}, \mathfrak{q}_{\text{aceita}}, \mathsf{Y}\right)$, se $\delta\left(\alpha_0, \mathfrak{q}_0\right) = \left(\alpha_r, \mathfrak{q}_{j_1}, \mathsf{Y}\right)$.

14. $\delta_{[2]}\left(\gamma_0, \mathfrak{b}\right) = \left(\lambda_{h(r)}, \mathfrak{q}_{\text{rejeita}}, \mathsf{Y}\right)$, se $\delta\left(\alpha_0, \mathfrak{q}_0\right) = \left(\alpha_r, \mathfrak{q}_{j_2}, \mathsf{Y}\right)$.

15. $\delta_{[2]}\left(\beta_0, \mathfrak{b}\right) = \left(\beta_{m,0,-,\mathsf{L}}, \mathfrak{a}, \mathsf{L}\right)$.

16. $\delta_{[2]}\left(\gamma_{i'}, \mathfrak{a}\right) = \left(\rho_{i'}, \mathfrak{b}, \mathsf{L}\right)$, se $\delta\left(\alpha_{i'}, \mathfrak{q}_0\right) = \left(\alpha_r, \mathfrak{q}_J, \mathsf{Y}\right)$.

17. $\delta_{[2]}\left(\rho_{i'}, \mathfrak{a}\right) = \left(\beta_{i'}, \mathfrak{q}_{\text{rejeita}}, \mathsf{R}\right)$.

18. $\delta_{[2]}\left(\rho_{i'}, \mathfrak{b}\right) = \left(\lambda_{h(r),J,+,\mathsf{Y}}, \mathfrak{b}, \mathsf{Y}\right)$, se $\delta\left(\alpha_{i'}, \mathfrak{q}_0\right) = \left(\alpha_r, \mathfrak{q}_J, \mathsf{Y}\right)$.

19. $\delta_{[2]}\left(\gamma_{i'}, \mathfrak{a}\right) = \left(\pi_{i'}, \mathfrak{b}, \mathsf{L}\right)$, se $\delta\left(\alpha_{i'}, \mathfrak{q}_0\right) = \left(\alpha_r, \mathfrak{q}_{j_1}, \mathsf{Y}\right)$ ou $\delta\left(\alpha_{i'}, \mathfrak{q}_0\right) = \left(\alpha_r, \mathfrak{q}_{j_2}, \mathsf{Y}\right)$.

20. $\delta_{[2]}\left(\pi_{i'}, \mathfrak{a}\right) = \left(\beta_{i'}, \mathfrak{q}_{\text{rejeita}}, \mathsf{R}\right)$.

21. $\delta_{[2]}\left(\pi_{i'}, \mathfrak{b}\right) = \left(\lambda_{h(r)}, \mathfrak{q}_{\text{aceita}}, \mathsf{Y}\right)$, se $\delta\left(\alpha_{i'}, \mathfrak{q}_0\right) = \left(\alpha_r, \mathfrak{q}_{j_1}, \mathsf{Y}\right)$.

22. $\delta_{[2]}\left(\pi_{i'}, \mathfrak{b}\right) = \left(\lambda_{h(r)}, \mathfrak{q}_{\text{rejeita}}, \mathsf{Y}\right)$, se $\delta\left(\alpha_{i'}, \mathfrak{q}_0\right) = \left(\alpha_r, \mathfrak{q}_{j_2}, \mathsf{Y}\right)$.

23. $\delta_{[2]}\left(\gamma_{i'}, \mathfrak{b}\right) = \left(\beta_{i',0,-,\mathsf{L}}, \mathfrak{a}, \mathsf{L}\right)$.

24. $\delta_{[2]}\left(\lambda_{h(i)}, \mathfrak{q}\right) = \left(\lambda_{h(i),0,-,\mathsf{R}}, \mathfrak{a}, \mathsf{R}\right)$.

25. $\delta_{[2]}\left(\lambda_{h(i),j,+,y}, \mathfrak{b}\right) = \left(\lambda_{h(r),J,+,\mathsf{Y}}, \mathfrak{b}, \mathsf{Y}\right)$, se $\delta\left(\alpha_i, \mathfrak{q}_j\right) = \left(\alpha_r, \mathfrak{q}_J, \mathsf{Y}\right)$.

26. $\delta_{[2]}\left(\lambda_{h(i),j,+,y}, \mathfrak{b}\right) = \left(\lambda_{h(r)}, \mathfrak{q}_{\text{aceita}}, \mathsf{Y}\right)$, se $\delta\left(\alpha_i, \mathfrak{q}_j\right) = \left(\alpha_r, \mathfrak{q}_{j_1}, \mathsf{Y}\right)$.

27. $\delta_{[2]}\left(\lambda_{h(i),j,+,y}, \mathfrak{b}\right) = \left(\lambda_{h(r)}, \mathfrak{q}_{\text{rejeita}}, \mathsf{Y}\right)$, se $\delta\left(\alpha_i, \mathfrak{q}_j\right) = \left(\alpha_r, \mathfrak{q}_{j_2}, \mathsf{Y}\right)$.

28. $\delta_{[2]}\left(\lambda_{h(i),j'+1,+,y}, \mathfrak{a}\right) = \left(\lambda_{h(i),j',+,y}, \mathfrak{b}, \mathsf{R}\right)$.

29. $\delta_{[2]}\left(\lambda_{h(i),0,+,y}, \mathfrak{a}\right) = \left(\lambda_{h(i)}, \mathfrak{a}, \mathsf{R}\right)$.

30. $\delta_{[2]}\left(\lambda_{h(i),j',-,y}, \mathfrak{b}\right) = \left(\lambda_{h(i),j'+1,-,y}, \mathfrak{a}, y\right)$.

31. $\delta_{[2]}\left(\lambda_{h(i),j,-,y}, \mathfrak{a}\right) = \left(\lambda_{h(r),J,+,\mathsf{Y}}, \mathfrak{b}, \mathsf{Y}\right)$, se $\delta\left(\alpha_i, \mathfrak{q}_j\right) = \left(\alpha_r, \mathfrak{q}_J, \mathsf{Y}\right)$.

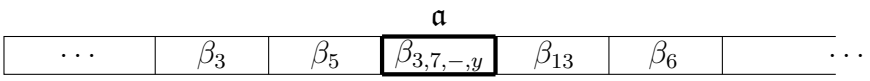
32. $\delta_{[2]}\left(\lambda_{h(i),j,-,y}, \mathfrak{a}\right) = \left(\lambda_{h(r)}, \mathfrak{q}_{\text{aceita}}, \mathsf{Y}\right)$, se $\delta\left(\alpha_i, \mathfrak{q}_j\right) = \left(\alpha_r, \mathfrak{q}_{j_1}, \mathsf{Y}\right)$.

33. $\delta_{[2]}\left(\lambda_{h(i),j,-,y}, \mathfrak{a}\right) = \left(\lambda_{h(r)}, \mathfrak{q}_{\text{rejeita}}, \mathsf{Y}\right)$, se $\delta\left(\alpha_i, \mathfrak{q}_j\right) = \left(\alpha_r, \mathfrak{q}_{j_2}, \mathsf{Y}\right)$.

As demais transições ficam livres, pois não interferirão na computação da máquina.

Comecemos mostrando como usamos as operações de salto para, com apenas $\mathfrak{a}$ e $\mathfrak{b}$, irmos fazendo trabalho análogo ao que faria a máquina $\mathcal{M}$. Para isso, suponhamos que, sendo $7, 5 \notin \{j_1, j_2\}$, então $\delta\left(\alpha_3, \mathfrak{q}_7\right) = \left(\alpha_8, \mathfrak{q}_5, \mathsf{R}\right)$. Eis como é o procedimento com a máquina $\mathcal{M}_{[2]}$.

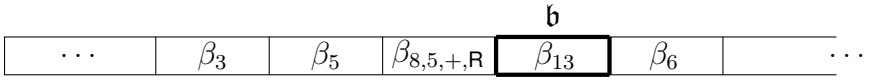
Figura 56 – Exemplo de operação de salto na máquina $\mathcal{M}_{[2]}$ (parte 1)



Fonte: Autor (2024)

Deixamos o estado atual da máquina sendo indicado acima do quadrado escaneado. A próxima configuração se dá a partir do item 6 da função de transição $\delta_{[2]}$.

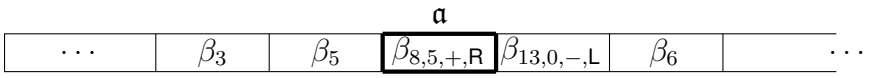
Figura 57 – Exemplo de operação de salto na máquina $\mathcal{M}_{[2]}$ (parte 2)



Fonte: Autor (2024)

Em seguida, a próxima configuração se dá por causa do item 2 da transição.

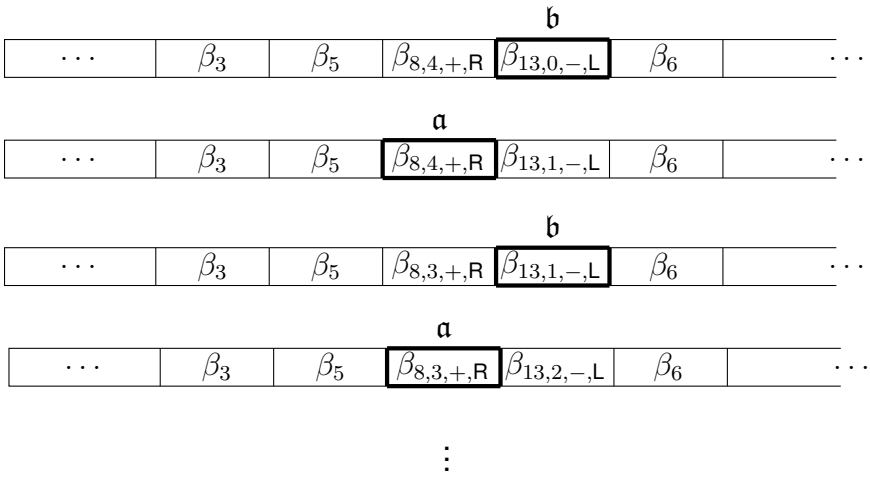
Figura 58 – Exemplo de operação de salto na máquina $\mathcal{M}_{[2]}$ (parte 3)



Fonte: Autor (2024)

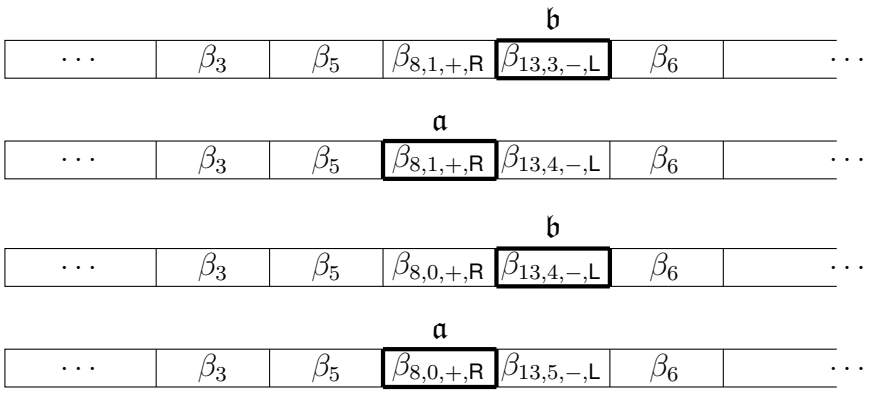
E depois dela, a sequência de transições que veremos se dá por alternância entre os itens 3 e 5 da transição.

Figura 59 – Exemplo de operação de salto na máquina $\mathcal{M}_{[2]}$ (parte 4)



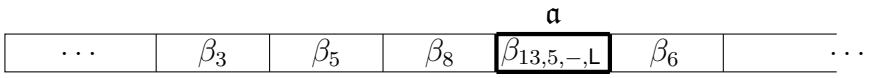
Fonte: Autor (2024)

Figura 60 – Exemplo de operação de salto na máquina $\mathcal{M}_{[2]}$ (parte 5)



Fonte: Autor (2024)

Figura 61 – Exemplo de operação de salto na máquina $\mathcal{M}_{[2]}$ (parte 6)

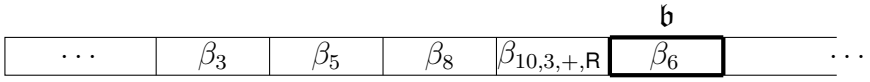


Fonte: Autor (2024)

E enfim, temos a configuração obtida devido ao item 4.

Depois disto, dependerá se, em $\mathcal{M}$, o próximo movimento da cabeça seria para a esquerda ou para a direita. Digamos que $3 \notin \{j_1, j_2\}$. Se tivéssemos $\delta(\alpha_{13}, \mathfrak{q}_5) = (\alpha_{10}, \mathfrak{q}_3, \mathsf{R})$, então $\mathcal{M}_{[2]}$ agiria a partir do item 6.

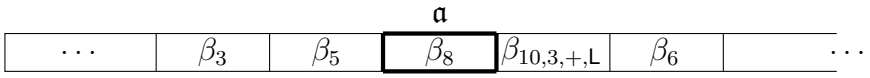
Figura 62 – Continuação do exemplo com transição comum para a direita



Fonte: Autor (2024)

E a operação de salto seria semelhante à que acabamos de realizar. Mas se, ao contrário, tivéssemos $\delta(\alpha_{13}, \mathfrak{q}_5) = (\alpha_{10}, \mathfrak{q}_3, \mathsf{L})$, a nossa máquina agiria a partir do item 7. A próxima configuração seria, na verdade, como a seguir.

Figura 63 – Continuação do exemplo com transição comum para a esquerda



Fonte: Autor (2024)

Teríamos, então, outra operação de salto, mas, desta vez, com o símbolo antigo à direita transmitindo a próxima configuração para a esquerda.

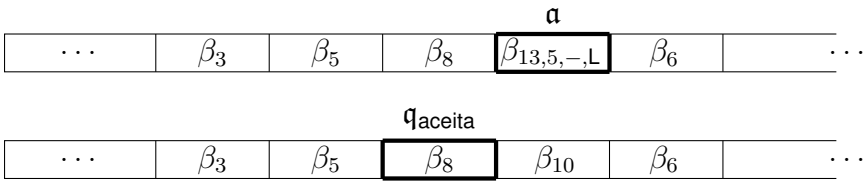
Estes procedimentos com os primeiros 7 itens da descrição de $\delta_{[2]}$ englobam toda a construção da nova máquina na bibliografia. Isto porque o autor não lidava

com três coisas que estão sendo tratadas aqui: alfabeto de entrada, fita com início e estados de parada. Por isso, originalmente, a máquina idealizada podia contar com apenas dois estados.

No entanto, para que lidemos com uma máquina de Turing segundo a definição que apresentamos, devemos reelaborar algumas coisas. Já é possível ver quais outros itens dedicamos à questão dos estados de parada.

Retomando o exemplo antes visto, digamos que $\delta(\alpha_{13}, \mathfrak{q}_5) = (\alpha_{10}, \mathfrak{q}_3, \mathsf{L})$, mas, dessa vez, que $j_1 = 3$. Neste caso, recorreríamos ao item 8.

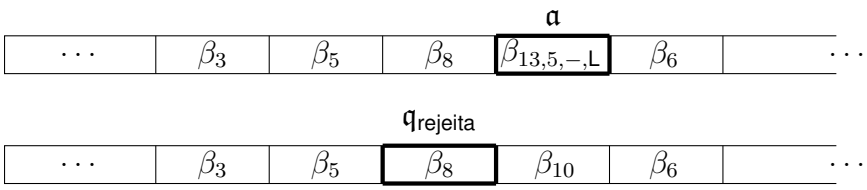
Figura 64 – Continuação do exemplo, com aceitação



Fonte: Autor (2024)

E nossa máquina, então, aceitaria esta entrada, semelhantemente a como aceitou a entrada associada a ela na máquina original. E, caso ainda tivéssemos que $\delta(\alpha_{13}, \mathfrak{q}_5) = (\alpha_{10}, \mathfrak{q}_3, \mathsf{L})$, porém $j_2 = 3$, acabaríamos recorrendo ao item 9.

Figura 65 – Continuação do exemplo, com rejeição



Fonte: Autor (2024)

E rejeitamos tal entrada. Mas, justamente nesta questão dos estados de parada haverá uma diferença marcante entre $\mathcal{M}$ e $\mathcal{M}_{[2]}$.

Todas as entradas aceitas por $\mathcal{M}_{[2]}$ serão entradas associadas a alguma entrada aceita por $\mathcal{M}$. E todas entradas aceitas por $\mathcal{M}$ estão sempre associadas a alguma entrada aceita por $\mathcal{M}_{[2]}$.

Da mesma maneira se dá quanto a entrar em loop. Todas as entradas de $\mathcal{M}_{[2]}$ que entram em loop estão associadas a alguma entrada de $\mathcal{M}$ que também entra em loop. E a recíproca também vale. Isto é, todas as entradas de $\mathcal{M}$ que entram em loop estão sempre associadas a alguma entrada de $\mathcal{M}_{[2]}$ que entra em loop.

Só que há entradas rejeitadas por $\mathcal{M}_{[2]}$ que não estão associadas a nenhuma entrada de $\mathcal{M}$. Por sua vez, todas as entradas rejeitadas por $\mathcal{M}$ estão sim sempre associadas a alguma entrada rejeitada por $\mathcal{M}_{[2]}$.

Mas o que queremos dizer com esta “associação”? Do modo como faremos, cada cadeia de entrada de $\mathcal{M}$ possui pelo menos uma cadeia de entrada cuja conclusão é correlacionada no que diz respeito à saída em $\mathcal{M}_{[2]}$, tanto em questão de símbolos como em questão do estado de parada.

E nenhuma cadeia de $\mathcal{M}_{[2]}$ está associada a mais de uma cadeia de $\mathcal{M}$. Relembremos que temos como alfabeto de entrada $\Sigma_{[2]} = \{\beta_1, \beta_2, \ldots, \beta_{m'}, \gamma_1, \gamma_2, \ldots, \gamma_{m'}\}$.

Para a cadeia vazia, a única entrada associada é a própria cadeia vazia. A cadeia α_{i_1} está associada à cadeia γ_{i_1}. E, com $k > 1$ natural, a cadeia $\alpha_{i_1}\alpha_{i_2}\alpha_{i_3}\ldots\alpha_{i_k}$ está associada às cadeias $\gamma_{i_1}u_2u_3\ldots u_k$ de modo que $u_2 \in \{\beta_{i_2}, \gamma_{i_2}\}$, $u_3 \in \{\beta_{i_3}, \gamma_{i_3}\}$, ..., e $u_k \in \{\beta_{i_k}, \gamma_{i_k}\}$.

Isto é, uma cadeia de comprimento k em $\mathcal{M}$ terá 2^{k-1} cadeias respectivas em $\mathcal{M}_{[2]}$. Por exemplo, a cadeia de entrada $\alpha_2\alpha_4\alpha_5$ está associada a 4 cadeias na nova máquina: $\gamma_2\beta_4\beta_5$, $\gamma_2\beta_4\gamma_5$, $\gamma_2\gamma_4\beta_5$ e $\gamma_2\gamma_4\gamma_5$.

As demais cadeias da nova máquina não estão associadas a nenhuma cadeia da máquina original. São estas exatamente todas aquelas entradas que iniciam por um símbolo $\beta_{i'}$, e, como verificaremos, serão todas rejeitadas.

Na bibliografia original, estava determinado que o primeiro símbolo das cadeias de entrada fosse da forma $\beta_{i',j,-,y}$ (na verdade da forma $\beta_{i,j,-,y}$). Mas, como antes mencionamos, o autor não estava trabalhando com o conceito de alfabeto de entrada. E, se adotássemos estes símbolos no nosso alfabeto de entrada, visivelmente a computação não seria satisfatória em entradas cujos símbolos posteriores ao inicial fossem estes. As operações de salto com eles ficariam comprometidas.

Como estes símbolos já possuem uma definição, na função de transição, que não é compatível com algo restrito ao início da fita, decidimos criar símbolos exclusivos para o início da fita. Estes primeiros símbolos são os $\gamma_{i'}$. Estes, junto dos $\beta_{i'}$ compõem todo o alfabeto de entrada.

No fundo, um símbolo $\gamma_{i'}$ não significa somente um correspondente ao símbolo $\alpha_{i'}$, mas também à configuração $(\alpha_{i'}, \mathfrak{q}_0)$ localizada no primeiro quadrado da fita. Se será a primeira situação ou a segunda, dependerá da posição em que ele se encontra na entrada e de como tal cadeia começa.

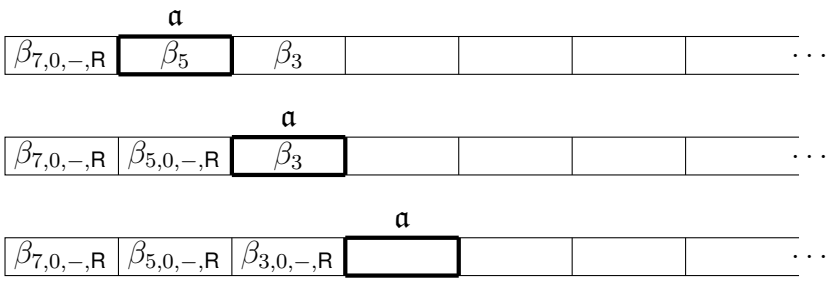
Tomemos, primeiramente, uma amostra do porquê de não podermos, da maneira como foi definida a transição, começar com uma cadeia que possua apenas símbolos $\beta_{i'}$, se tivermos a intenção de replicar o trabalho da máquina original.

Figura 66 – Amostra da computação de entradas com apenas símbolos $\beta_{i'}$ (parte 1)

Fonte: Autor (2024)

E o que veremos aqui é uma sequência do uso do item 1.

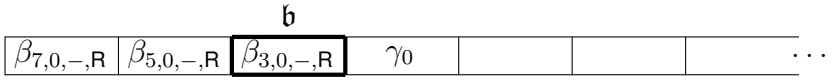
Figura 67 – Amostra da computação de entradas com apenas símbolos $\beta_{i'}$ (parte 2)



Fonte: Autor (2024)

Quando a cabeça chega ao primeiro símbolo $\beta_0 = \text{␣}$, devemos partir do item 10.

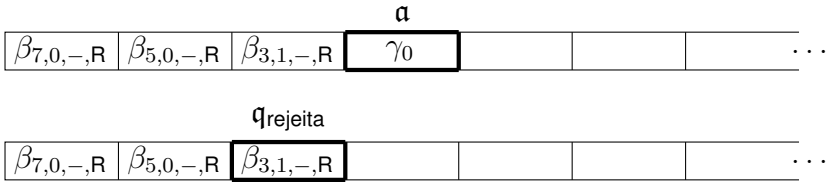
Figura 68 – Amostra da computação de entradas com apenas símbolos $\beta_{i'}$ (parte 3)



Fonte: Autor (2024)

Depois, passamos pelo item 5, seguido do item 11.

Figura 69 – Amostra da computação de entradas com apenas símbolos $\beta_{i'}$ (parte 4)



Fonte: Autor (2024)

Tal como neste caso, entradas assim sempre serão rejeitadas, não importando o que teria acontecido na máquina original com os símbolos análogos. Aquilo que o autor definira nos levaria a sempre deixar símbolos da forma $\beta_{i,j,-,\mathrm{R}}$ no caminho e nunca completar uma operação de salto.

Aquilo que aconteceu quando a cabeça encontrou o primeiro símbolo vazio foi fruto de nossa definição. Colocamos o símbolo γ_0 justamente para que a máquina cheque se o antigo símbolo vazio estava no início da fita ou não. Se estava, o símbolo γ_0 vai acabar recebendo o estado $\mathfrak{b}$, e prosseguir com a computação. Se não estava, o símbolo γ_0 vai receber o estado $\mathfrak{a}$ e rejeitar a entrada.

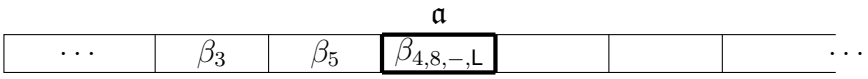
Esta definição foi pensada tanto para este caso (de uma entrada só com símbolos da forma $\beta_{i'}$) quanto para a ocasião em que há a entrada vazia, visto que em ambas o espaço vazio é lido conjuntamente com o estado $\mathfrak{a}$. Em todas as outras situações, o símbolo $\beta_0 = \text{␣}$ será escaneado com o estado $\mathfrak{b}$ (sem contar os estados de

parada).

Afinal, quando a entrada não é vazia, o espaço vazio estará sempre à direita dos outros símbolos. Quando o último símbolo da cadeia for iniciar a operação de salto com o quadrado vazio à sua direita, este quadrado vazio receberá, com a cabeça, o estado $\mathfrak{b}$, responsável por levar a cabeça de volta para a esquerda a fim de seguir com tal operação. E assim também será com os outros espaços vazios após o fim da fita.

Exemplificamos isto abaixo, supondo, para tal, que $\delta\left(\alpha_4, \mathfrak{q}_8\right) = \left(\alpha_2, \mathfrak{q}_4, \mathsf{R}\right)$, com $4 \notin \{j_1, j_2\}$.

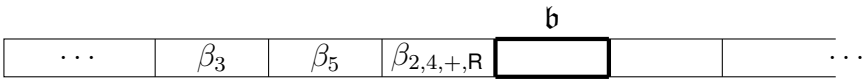
Figura 70 – Exemplo de operação de salto envolvendo o ␣ (parte 1)



Fonte: Autor (2024)

Depois, passamos pela transição dada a partir do item 6.

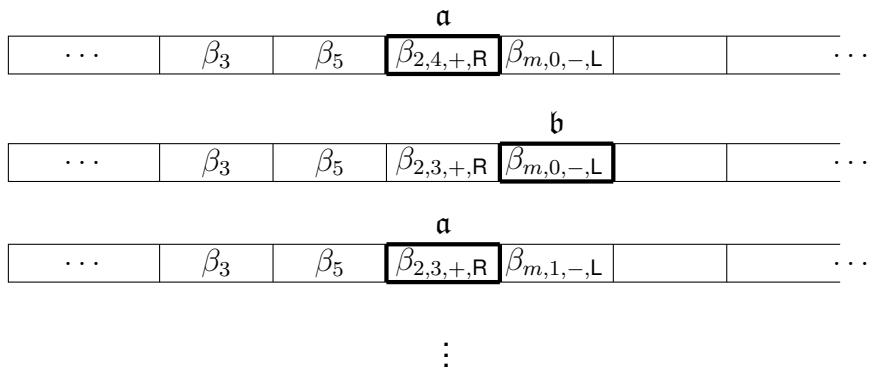
Figura 71 – Exemplo de operação de salto envolvendo o ␣ (parte 2)



Fonte: Autor (2024)

E, pelo item 15, teremos conforme na sequência. Depois o que vem é a alternância entre os itens que constituem uma operação de salto.

Figura 72 – Exemplo de operação de salto envolvendo o ␣ (parte 3)

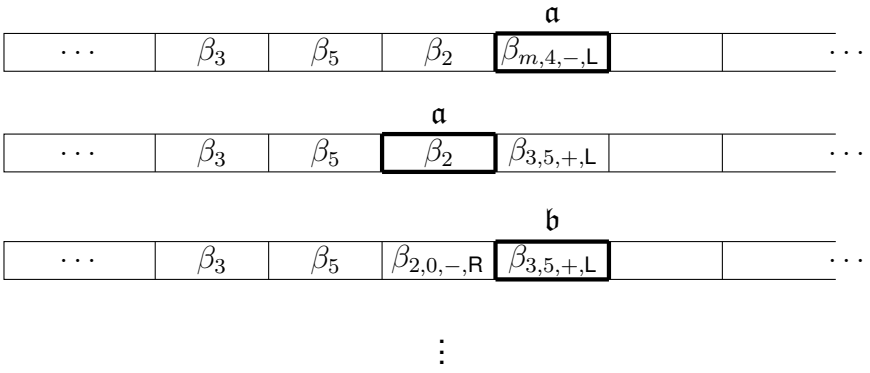


Fonte: Autor (2024)

Mas do que se trata este beta com um índice m para os símbolos? Explicamos. Quanto ao espaço vazio, ainda poderíamos levantar uma objeção. A de que o espaço vazio, para replicar o trabalho da máquina original, é passível de aparecer, impresso,

em outros lugares da fita. E justamente para isso que produzimos o símbolo β_m. Digamos que $\delta\left(\alpha_0, \mathfrak{q}_4\right) = (\alpha_3, \mathfrak{q}_5, \mathsf{L})$, com $5 \notin \{j_1, j_2\}$.

Figura 73 – Continuação do exemplo



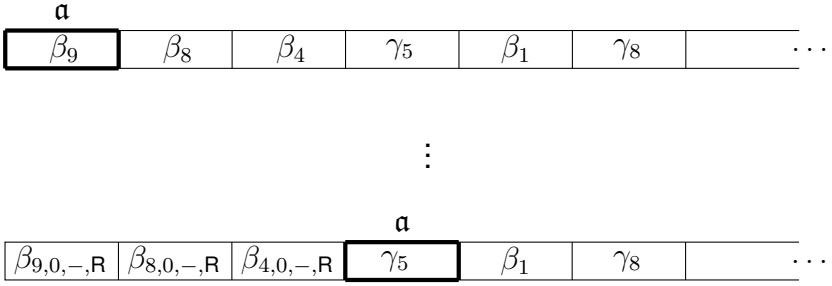
Fonte: Autor (2024)

O símbolo β_m ocupará o lugar que viria a ser de um espaço vazio quando impresso no quadrado correspondente de $\mathcal{M}$. E o símbolo $\beta_m = \beta_{h(0)}$ já possui a mesma maneira de alavancar a transição vista de início. Ou seja, usamos, com este símbolo, os mesmos primeiros itens que definem a função de transição da nossa máquina $\mathcal{M}_{[2]}$, os quais antes exploramos.

Conseguimos perceber que a computação da máquina de Turing vai continuar como planejado. Olhemos, agora, para as cadeias que começam com um símbolo pertencente ao conjunto $\{\beta_1, \beta_2, \ldots, \beta_{m'}\}$, mas que possuem algum símbolo da forma $\gamma_{i'}$. Uma cadeia de entrada dessas persiste no problema de deixar pelo caminho símbolos que deveriam servir para as operações de salto, até que chegue ao símbolo $\gamma_{i'}$.

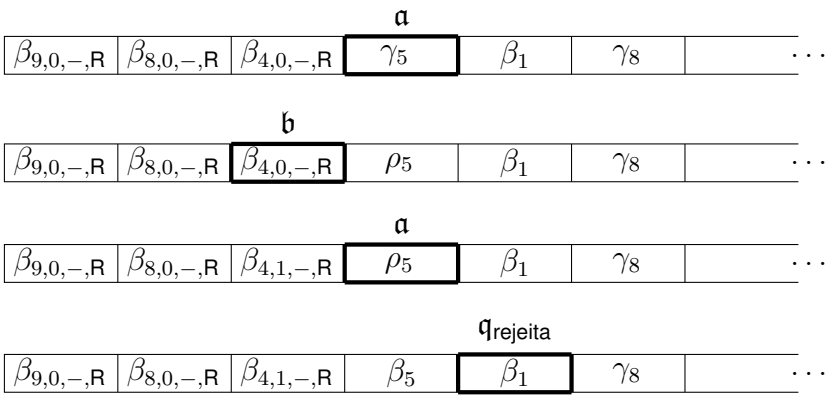
Para exemplificarmos, digamos que haja a transição $\delta\left(\alpha_5, \mathfrak{q}_0\right) = (\alpha_6, \mathfrak{q}_4, \mathsf{R})$ e $4 \notin \{j_1, j_2\}$.

Figura 74 – Amostra do primeiro caso de computação de entradas que iniciam em símbolos $\beta_{i'}$ e possuem símbolos $\gamma_{i'}$ (parte 1)



Fonte: Autor (2024)

Figura 75 – Amostra do primeiro caso de computação de entradas que iniciam em símbolos $\beta_{i'}$ e possuem símbolos $\gamma_{i'}$ (parte 2)

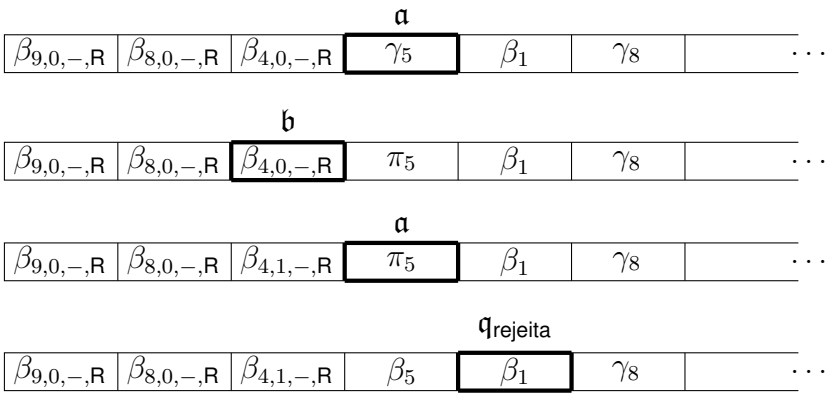


Fonte: Autor (2024)

Aqui usamos, desde que chegamos em γ_5, respectivamente os itens 16, 5 e 17 da transição.

Só que ainda havemos de ver o que aconteceria se $\delta(\alpha_5, \mathfrak{q}_0) = (\alpha_6, \mathfrak{q}_4, \mathsf{R})$ mas $4 \in \{j_1, j_2\}$. Este caso aqui também terminará em rejeição.

Figura 76 – Amostra do segundo caso de computação de entradas que iniciam em símbolos $\beta_{i'}$ e possuem símbolos $\gamma_{i'}$



Fonte: Autor (2024)

Nesta situação, o que acabamos usando são os itens 19, 5 e 20 da transição, respectivamente. Com isto, fica visto como todas as entradas deste formato irão acabar em rejeição.

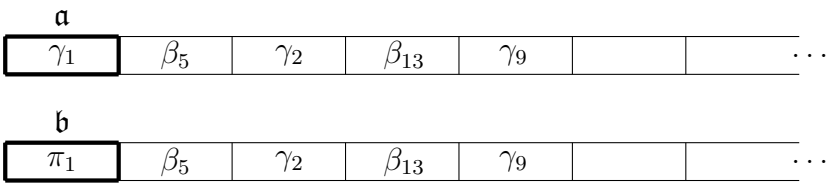
Este último caso apresentado parece indiferente. Acontece que ele foi pensado para uma situação em que esta distinção importa. Trata-se de um caso que ocorre exatamente no início da fita.

Dado isso, voltemos nossos olhos para algo ainda não explorado: o início de entradas associadas a entradas de $\mathcal{M}$.

Ou seja, tomemos exemplos em que as entradas comecem com um símbolo da forma $\gamma_{i'}$. Para isso, comecemos supondo que $\delta(\alpha_1, \mathfrak{q}_0) = (\alpha_9, \mathfrak{q}_3, \mathsf{L})$, e vejamos já o

que ocorreria se $3 \in \{j_1, j_2\}$. Recorremos, então, ao item 19 da função de transição mais uma vez.

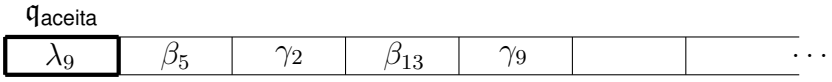
Figura 77 – Exemplo do primeiro caso de um $\gamma_{i'}$ inicial sobre o estado $\mathfrak{a}$



Fonte: Autor (2024)

Vejamos o que se dá se $3 = j_1$. Utilizamos o item 21.

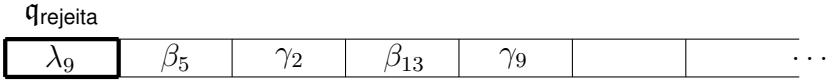
Figura 78 – Exemplo do primeiro caso com aceitação



Fonte: Autor (2024)

Enquanto que, se $3 = j_2$, utilizamos o item 22.

Figura 79 – Exemplo do primeiro caso com rejeição



Fonte: Autor (2024)

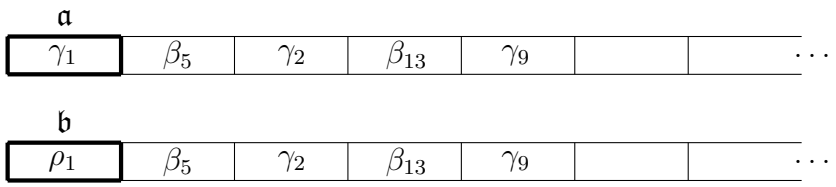
É aqui que se encontra a necessidade da distinção. Pois devemos conseguir fazer algo para que a máquina $\mathcal{M}_{[2]}$ aceite imediatamente a entrada caso $\mathcal{M}$ já aceitasse imediatamente a entrada associada. E também fazer o mesmo quando remetesse à questão da rejeição.

Como ocorre aqui, e sempre ocorrerá, reservamos os símbolos $\lambda_{h(i)}$ para serem os símbolos que representam o que haveria no primeiro quadrado da máquina original. Em verdade, os símbolos $\gamma_{i'}$ servem apenas para a entrada, pois quando escaneados, serão substituídos sempre por símbolos de outro formato, e nunca substituem outros símbolos no quadrado escaneado.

Como confirmaremos, no caso das entradas de $\mathcal{M}_{[2]}$ associadas a entradas de $\mathcal{M}$, os símbolos $\gamma_{i'}$, quando não no início da entrada, acabam sendo, depois, substituídos por símbolos $\beta_{h(i)}$.

Lidemos novamente com a transição $\delta(\alpha_1, \mathfrak{q}_0) = (\alpha_9, \mathfrak{q}_3, \mathsf{L})$. Só que dessa vez, a veremos a partir da situação onde $3 \notin \{j_1, j_2\}$. Recorremos, então, ao item 16 da função mais uma vez.

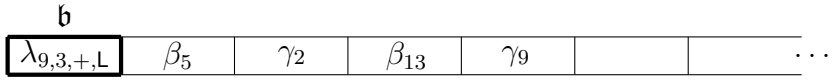
Figura 80 – Exemplo do segundo caso de um $\gamma_{i'}$ inicial sobre o estado $\mathfrak{a}$ (parte 1)



Fonte: Autor (2024)

Daí, usamos o item 18. Entendamos que os símbolos $\rho_{i'}$ servem para, quando a transição em δ não resultar em parada, verificar se o símbolo $\gamma_{i'}$ anterior estava ou não no início da fita. Prossigamos com o exemplo.

Figura 81 – Exemplo do segundo caso de um $\gamma_{i'}$ inicial sobre o estado $\mathfrak{a}$ (parte 2)

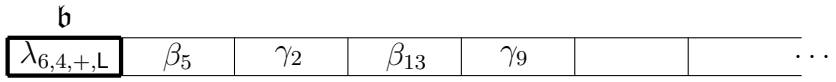


Fonte: Autor (2024)

Notemos que, caso a transição em δ levasse a cabeça original para a direita em vez da esquerda, ainda pelo item 18, o primeiro quadrado, após a transição em $\delta_{[2]}$ seria preenchido com $\lambda_{9,3,+,\mathsf{R}}$. Ainda nos encontraríamos no estado $\mathfrak{b}$ como atual, e estaríamos escaneando o segundo quadrado da fita, pois a cabeça andou para a direita.

Mas demos continuidade ao exemplo já graficamente exposto. Diante do símbolo escaneado $\lambda_{9,3,+,\mathsf{L}}$ e do estado $\mathfrak{b}$, digamos que $\delta\left(\alpha_9, \mathfrak{q}_3\right) = \left(\alpha_6, \mathfrak{q}_4, \mathsf{L}\right)$, com $4 \notin \{j_1, j_2\}$. Recorremos, então, ao item 25 da transição.

Figura 82 – Continuação do exemplo do segundo caso (parte 1)



Fonte: Autor (2024)

Diferente do que acontece com os símbolos da forma $\beta_{i,j'+1,+,y}$, quando um símbolo da forma $\lambda_{i,j'+1,+,y}$ é escaneado com o estado $\mathfrak{b}$, ele não será substituído na ordem decrescente do índice relativo aos estados de $\mathcal{M}$.

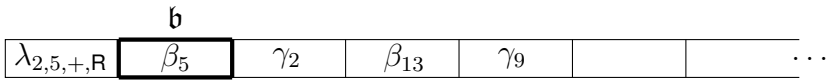
Isso porque este estado se encontra presente durante o escaneamento deste símbolo quando tal símbolo sinalizava uma transmissão: a transmissão de informação na operação de salto. A cabeça tentou se mover para a esquerda em direção ao símbolo que supostamente receberia tal informação. Mas, como se trata do início da fita, estamos no mesmo quadrado.

Então o que nós faremos é já substituir o símbolo de $\mathcal{M}_{[2]}$ relativo à antiga transmissão de um estado a partir de um símbolo (ambos de $\mathcal{M}$) pelo símbolo de $\mathcal{M}_{[2]}$ relativo à próxima transmissão de um estado a partir de um símbolo (de $\mathcal{M}$, novamente).

Caso tivéssemos $4 = j_1$, o que aconteceria, segundo o item 26, seria simplesmente a impressão de λ_6 no início da fita e, em seguida, a máquina entraria no estado de aceitação. E se tivéssemos $4 = j_2$, segundo o item 27, a máquina faria apenas a impressão de λ_6 no início da fita com a sucessão para o estado de rejeição.

Mas, continuando o exemplo, estamos novamente sobre um símbolo da forma $\lambda_{i,j,+,y}$ com o estado $\mathfrak{b}$. Dessa vez, digamos que $\delta(\alpha_6, \mathfrak{q}_4) = (\alpha_2, \mathfrak{q}_5, \mathsf{R})$, com $5 \notin \{j_1, j_2\}$. Ainda nos servimos do item 25 para esta transição, mas desta vez já sairemos do primeiro quadrado.

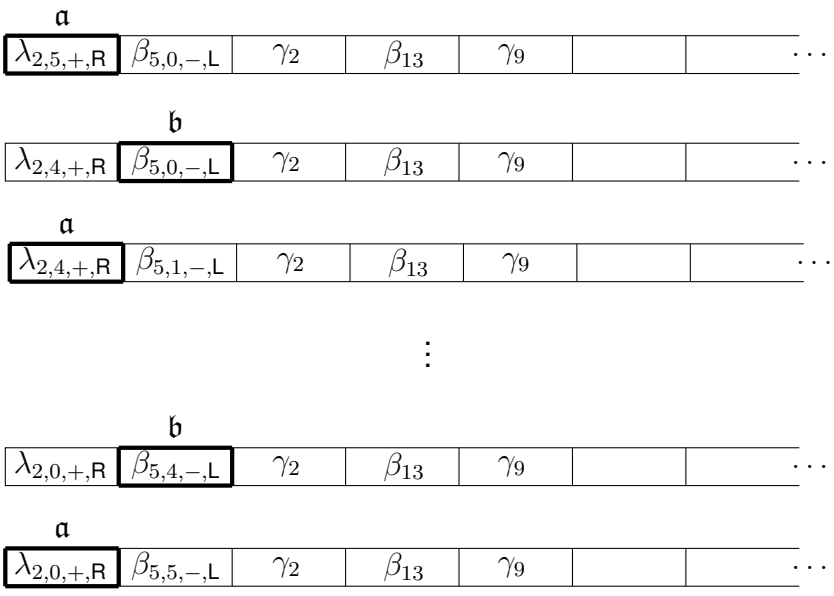
Figura 83 – Continuação do exemplo do segundo caso (parte 2)



Fonte: Autor (2024)

E a partir deste momento, o que presenciaremos dessa vez é uma operação de salto feita do início ao fim. Ela se dará com o primeiro quadrado, transmitindo a informação do estado $\mathfrak{q}_5$, para o segundo quadrado. Para isso, começamos com o item 2 da transição. Depois disso, nós vamos intercalando entre os itens 28 e 5 da definição.

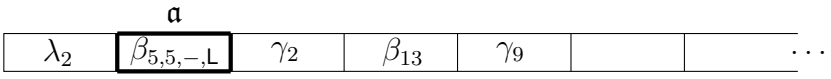
Figura 84 – Continuação do exemplo do segundo caso (parte 3)



Fonte: Autor (2024)

E daí, pelo item 29, finalmente conseguimos imprimir um símbolo no primeiro quadrado que não seja próprio de operações de salto. Deixamos ali um representante, para o início da fita, de um símbolo de $\mathcal{M}$. No exemplo em questão, este representante é o λ_2.

Figura 85 – Continuação do exemplo do segundo caso (parte 4)

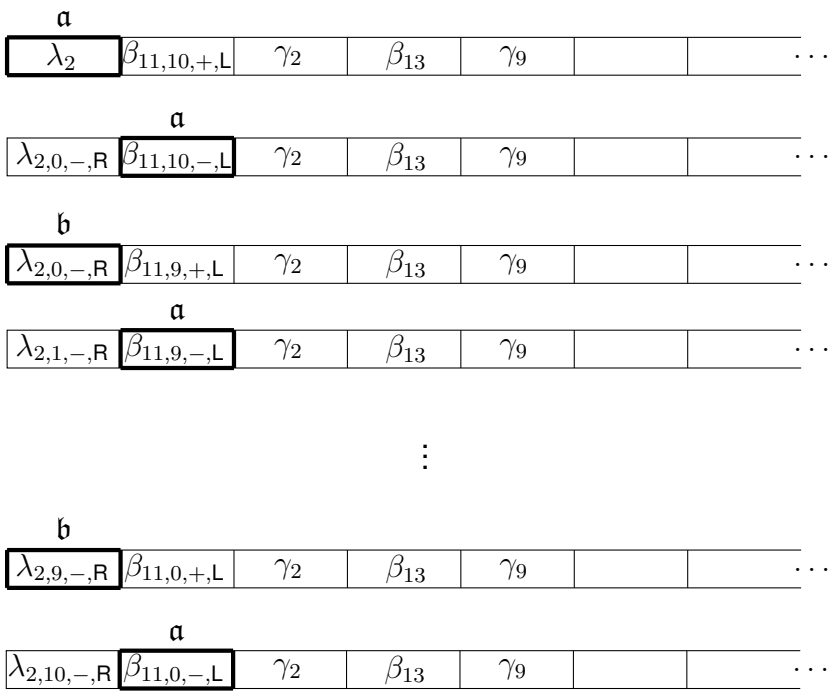


Fonte: Autor (2024)

E, com isso, a máquina $\mathcal{M}_{[2]}$ inicia sua computação. Mas, a partir deste exemplo que estamos tomando para nosso caso, ainda convém checar duas coisas: como o primeiro símbolo da fita (isto é, um λ_i) recebe a informação em uma operação de salto que começa à sua direita, e como os símbolos $\gamma_{i'}$ do meio destas entradas funcionam na sua computação.

Para verificarmos isto, digamos que $\delta(\alpha_5, \mathfrak{q}_5) = (\alpha_{11}, \mathfrak{q}_{10}, \mathsf{L})$, com $10 \notin \{j_1, j_2\}$. Começamos pelo item 6, seguido do item 24 da transição, e daí intercalaremos entre os itens 3 e 30.

Figura 86 – Exemplo de operação de salto envolvendo λ_i (parte 1)

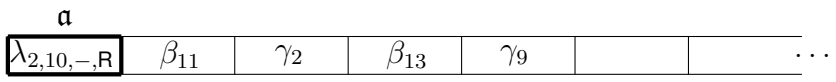


Fonte: Autor (2024)

Pelo item 4, nós vemos que estamos prestes a chegar ao fim da operação de salto.

Olhando para o que seria a configuração da máquina original neste momento,

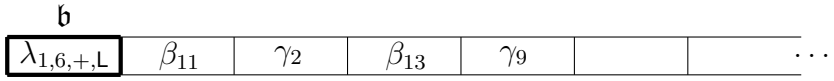
Figura 87 – Exemplo de operação de salto envolvendo λ_i (parte 2)



Fonte: Autor (2024)

digamos que $\delta\left(\alpha_2, \mathfrak{q}_1 0\right) = (\alpha_1, \mathfrak{q}_6, \mathsf{L})$, com $6 \notin \{j_1, j_2\}$. Pelo item 31 da função, teremos o seguinte.

Figura 88 – Exemplo de operação de salto envolvendo λ_i (parte 3)



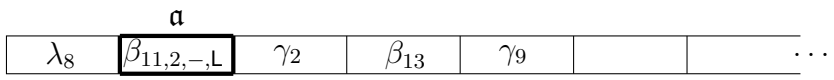
Fonte: Autor (2024)

E o procedimento diante desta configuração é algo que já conhecemos. Façamos rapidamente a referência ao que aconteceria caso tivéssemos $6 \in \{j_1, j_2\}$ na sequência.

No caso de $\mathfrak{q}_6$ ser o estado de aceitação, recorreríamos ao item 32 da função de transição e teríamos o símbolo λ_1 impresso no primeiro quadrado da fita, com o estado atual $\mathfrak{q}_{\text{aceita}}$. E no caso de $\mathfrak{q}_6$ ser o estado de rejeição, nós recorreríamos ao item 33, tendo o símbolo λ_1 sendo impresso no primeiro quadrado, com o estado atual $\mathfrak{q}_{\text{rejeita}}$.

Retornando ao exemplo como ele estava sendo proposto pelas figuras acima, avancemos a nossa computação em alguns passos, de modo que chegamos no estágio que representamos por primeiro abaixo.

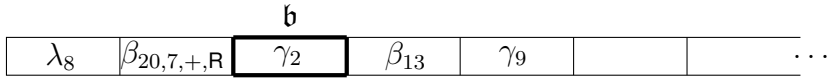
Figura 89 – Exemplo com γ_i presente no meio de uma entrada apropriada (parte 1)



Fonte: Autor (2024)

Agora, para vermos quanto àquele γ_2 de nosso exemplo, digamos que ocorra a transição $\delta\left(\alpha_{11}, \mathfrak{q}_2\right) = (\alpha_{20}, \mathfrak{q}_7, \mathsf{R})$, com $7 \notin \{j_1, j_2\}$. Pelo item 6 da função de transição, temos como adiante.

Figura 90 – Exemplo com γ_i presente no meio de uma entrada apropriada (parte 2)

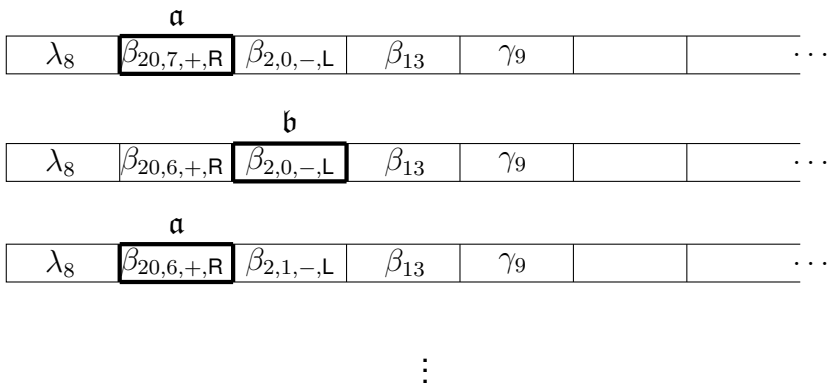


Fonte: Autor (2024)

E, como podemos ver através do item 23, prontamente teremos um símbolo do formato $\beta_{i',0,-,\mathsf{L}}$ onde estava o $\gamma_{i'}$. Fixamos esta definição pelo fato de que o único estado que os símbolos $\gamma_{i'}$ recebem quando estão em algum lugar fora do início de uma entrada é o estado $\mathfrak{b}$. Afinal eles estarão sempre à direita das antigas posições da cabeça de $\mathcal{M}_{[2]}$ e, quando escaneados, iniciarão uma operação de salto com o símbolo à sua esquerda.

Sendo assim, o que teremos aqui é uma operação de salto como todas as outras, seguindo sempre a lógica de replicar a computação de $\mathcal{M}$. Abaixo, sua representação.

Figura 91 – Exemplo com γ_i presente no meio de uma entrada apropriada (parte 3)



Fonte: Autor (2024)

Com isso, conseguimos observar que nossa máquina de Turing fará o trabalho a que se propõe com estas entradas. Dará sequência à computação delas, de modo que se a máquina original entrasse em loop sobre uma determinada entrada não-vazia, a máquina atual fará o mesmo sobre as entradas associadas a ela, e igualmente se tratarmos da aceitação e da rejeição.

Seja, então, $f_{[2]} : \Sigma^* \to \mathcal{P}\left(\Sigma^*_{[2]}\right)$ a função tal que $f_{[2]}(w) = A_w$, onde A_w é definido como sendo exatamente o conjunto de todas as cadeias de $\Sigma^*_{[2]}$ associadas a w do modo como descrevemos antes. Assim, $f_{[2]}$ é injetora, como também, para $w^{(1)}, w^{(2)} \in \Sigma^*$ com $w^{(1)} \neq w^{(2)}$, temos que $A_{w^{(1)}} \cap A_{w^{(2)}} = \varnothing$.

E ainda, conseguimos montar a nossa máquina $\mathcal{M}_{[2]}$ de tal forma que, para $z \in \Sigma^*_{[2]}$:

- se $\mathcal{M}$ aceita $w \in \Sigma^*$, então $\mathcal{M}_{[2]}$ aceita $z \in A_w$;
- se $\mathcal{M}$ entra em loop sobre $w \in \Sigma^*$, então $\mathcal{M}_{[2]}$ entra em loop sobre $z \in A_w$;
- se $\mathcal{M}$ rejeita $w \in \Sigma^*$, então $\mathcal{M}_{[2]}$ rejeita $z \in A_w$;
- se para todo $w \in \Sigma^*$ tivermos $z \notin A_w$, então $\mathcal{M}_{[2]}$ rejeita z.

Cabe ainda destacarmos uma coisa. As cadeias $\alpha_{i_1}\alpha_{i_2}\alpha_{i_3}\ldots\alpha_{i_k}$ formadas a partir do alfabeto da fita de $\mathcal{M}$, enquanto entradas, estão associadas às cadeias de entrada $\gamma_{i_1}u_2u_3\ldots u_k$ formadas a partir do alfabeto de entrada de $\mathcal{M}_{[2]}$, como já havíamos falado.

Todavia, para $K > 1$ natural, fora as entradas no início de computação, as cadeias $\alpha_{I_1}\alpha_{I_2}\alpha_{I_3}\ldots\alpha_{I_K}$ tomadas a partir do alfabeto da fita de $\mathcal{M}$ que aparecem durante a história de computação da máquina sobre uma entrada estão sempre associadas, já por meio de uma outra relação, às cadeias $\lambda_{h(I_1)}v_2v_3\ldots v_K$, tomadas a partir do alfabeto da fita de $\mathcal{M}_{[2]}$, onde

$$v_2 \in \begin{cases} \left\{\beta_{h(I_2)}, \gamma_{h(I_2)}\right\}, & \text{se } 1 \leq I_2 \leq m', \\ \left\{\beta_{h(I_2)}\right\}, & \text{se } I_2 = 0 \text{ ou } m' < I_2 \leq m \end{cases},$$

$$v_3 \in \begin{cases} \left\{\beta_{h(I_3)}, \gamma_{h(I_3)}\right\}, & \text{se } 1 \leq I_3 \leq m', \\ \left\{\beta_{h(I_3)}\right\}, & \text{se } I_3 = 0 \text{ ou } m' < I_3 \leq m \end{cases},$$

$$\vdots$$

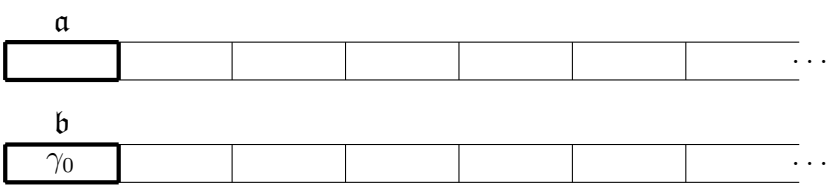
$$v_K \in \begin{cases} \left\{\beta_{h(I_K)}, \gamma_{h(I_K)}\right\}, & \text{se } 1 \leq I_K \leq m', \\ \left\{\beta_{h(I_K)}\right\}, & \text{se } I_K = 0 \text{ ou } m' < I_K \leq m \end{cases}.$$

Como também uma cadeia α_{I_1} que não na entrada, está associada à cadeia $\lambda_{h(I_1)}$. Neste quesito, a cadeia vazia na máquina original estará associada às cadeias λ_m e $\lambda_m\beta_m\beta_m\ldots\beta_m$.

Notemos que isto vale inclusive para as saídas de nossas máquinas $\mathcal{M}_{[2]}$. Ou seja, quando começamos com uma cadeia de entrada que satisfaz a primeira associação, ao terminarmos a computação, a cadeia de saída é tal que satisfaz a segunda associação.

E temos falado mais de uma vez a respeito da cadeia vazia dentro da fita. Vejamos, por fim, o que acontece com nossas máquinas de Turing diante de tal entrada. Para isto, suponhamos que ocorre a transição $\delta(\alpha_0, \mathfrak{q}_0) = (\alpha_2, \mathfrak{q}_6, \mathsf{L})$, primeiramente considerando $6 \notin \{j_1, j_2\}$. Olhamos, a partir disso, para o item 10 da função de transição de $\mathcal{M}_{[2]}$.

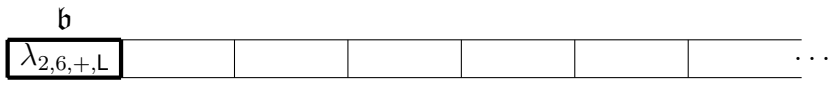
Figura 92 – Exemplo com a entrada vazia (parte 1)



Fonte: Autor (2024)

Pelo item 12, prosseguimos.

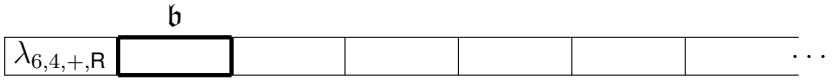
Figura 93 – Exemplo com a entrada vazia (parte 2)



Fonte: Autor (2024)

Depois disso, supondo que $\delta(\alpha_2, \mathfrak{q}_6) = (\alpha_6, \mathfrak{q}_4, \mathsf{R})$, com $4 \notin \{j_1, j_2\}$, segue o item 25.

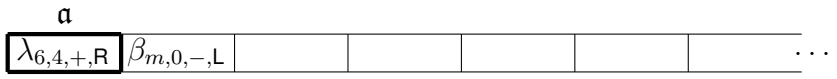
Figura 94 – Exemplo com a entrada vazia (parte 3)



Fonte: Autor (2024)

Daí prosseguimos com o item 15 de nossa função, como nós antes já havíamos visto.

Figura 95 – Exemplo com a entrada vazia (parte 4)

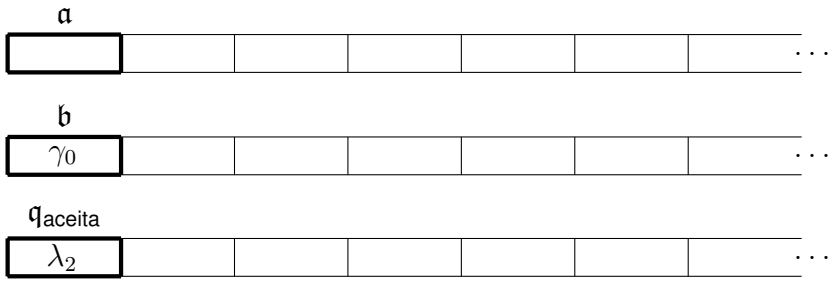


Fonte: Autor (2024)

Temos, a partir disto, uma operação de salto, e a computação segue normalmente.

Voltando à situação de $\delta(\alpha_0, \mathfrak{q}_0) = (\alpha_2, \mathfrak{q}_6, \mathsf{L})$, tenhamos agora que $6 = j_1$. Então, logo após o item 10, usamos o item 13.

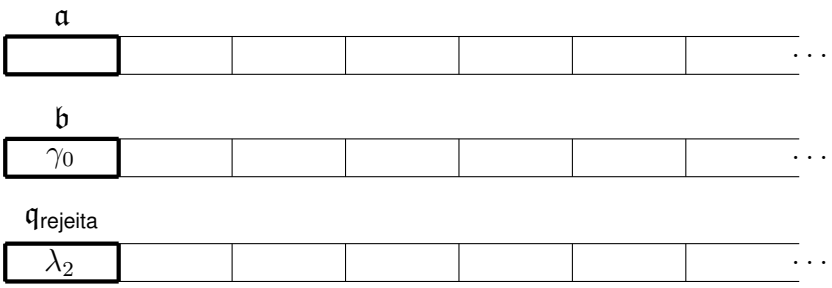
Figura 96 – Exemplo com a entrada vazia e aceitação imediata



Fonte: Autor (2024)

E, finalmente, mais uma vez nessa mesma transição em $\mathcal{M}$, tenhamos, desta vez, que $6 = j_2$. Passamos pelos itens 10 e 14.

Figura 97 – Exemplo com a entrada vazia e rejeição imediata



Fonte: Autor (2024)

Visto tudo isso, o que falávamos é claramente válido também para a entrada vazia de $\mathcal{M}$, associada à entrada vazia de $\mathcal{M}_{[2]}$.

Como havíamos antes falado, para os casos em que a máquina $\mathcal{M}$ já inicia com um estado de parada, não usamos esta mesma construção para $\mathcal{M}_{[2]}$. Podemos, em vez disso, usar o mesmo alfabeto de entrada Σ para $\mathcal{M}_{[2]}$, e ela terá apenas dois estados: o de aceitação e o de rejeição. Ou se quisermos, colocamos $\mathfrak{a}$ e $\mathfrak{b}$ também. Mas eles não farão diferença.

A máquina $\mathcal{M}_{[2]}$ não carecerá de nenhuma operação de salto. Tal nova máquina inicia com o estado de aceitação se $\mathcal{M}$ assim iniciava, e inicia com o estado de rejeição se $\mathcal{M}$ assim iniciava. Sendo assim, para estes casos, ou a linguagem de $\mathcal{M}_{[2]}$ será o próprio Σ^*, ou será então $\varnothing$.

E, fechando esta seção, finalmente podemos fazer uma importante afirmação. Se toda máquina de Turing pode ser replicada em uma máquina com apenas os quatro estados que apresentamos, então o podemos fazer inclusive com a máquina universal. Ou seja, existe a máquina $\mathcal{U}_{[2]}$.

6.2 PARA UMA MÁQUINA UNIVERSAL DE DOIS SÍMBOLOS

Dado o que foi falado nesta última seção, finalizando o trabalho baseado no mesmo autor, trazemos um resultado notável, que mais à frente será resgatado e aprofundado nesta dissertação: podemos montar uma máquina de Turing universal com apenas dois símbolos.

Um destes símbolos é o ␣, no lugar do qual o autor usa o símbolo 0. Este faz parte apenas do alfabeto da fita da máquina em questão. O outro símbolo é o 1, já constituindo o alfabeto de entrada. A ideia do autor era identificar as cadeias de uma máquina de Turing original com cadeias de sequências binárias em uma nova máquina que lhe replicasse o funcionamento.

Sendo $\mathcal{M}$ a máquina original, chamemos esta nova máquina de Turing de $\mathcal{M}^{[2]}$.

O autor, relembrando, não lidava com o conceito de alfabeto de entrada. E, desse modo, ele admitia que suas entradas tivessem os símbolos 0, isto é, os símbolos ␣.

Com m sendo a quantidade de símbolos do alfabeto da fita original, ele tomou o número l tal que fosse o menor natural para o qual $m \leq 2^l$. Assim, ele distribuiria os símbolos de $\mathcal{M}$, identificando-os com sequências binárias, cada uma em l quadrados da nova máquina. Começa com o símbolo ␣ identificado com a sequência de apenas símbolos 0, e vai identificando os demais na ordem numérica crescente.

Com uma série de estados para a nova máquina, ele analisa cada uma dessas sequências como se fosse a leitura de um símbolo escaneado em $\mathcal{M}$, enquanto carrega a informação daquele que seria o estado atual original. Depois disso, imprime a nova sequência no lugar, anda para a direita ou esquerda, e já carrega a informação do que seria o próximo estado. E segue indo de sequência em sequência.

Nesta seção, nós trataremos de como fazer isto funcionar, na questão da cadeia de entrada, com o 1 sendo o único símbolo do alfabeto de entrada. Primeiramente, identificaremos cada cadeia de símbolos 1 com um número natural. A cadeia 1 fica identificada com o número natural 0. A cadeia 11 fica identificada com o natural 1. A cadeia 111 com o natural 2. A cadeia 1111 com o natural 3. E assim por diante.

Sendo assim, o que faremos depois é identificar cada número natural com um par ordenado de um certo conjunto, utilizando método semelhante àquele com o que muitas vezes é verificada a enumerabilidade do conjunto dos números racionais. Para $y \in \mathbb{N} - \{0\}$, seja T_y o y-ésimo número triangular. Isto é, tome $T_y = 1 + 2 + 3 + \cdots + (y-1) + y$. Com isto, definimos a bijeção $\nu : (\mathbb{N} - \{0\}) \times \mathbb{N} \to \mathbb{N}$ tal que

$$\nu(L, M) = T_{L+M} - M - 1.$$

A tabela abaixo ilustra esta função.

Tabela 54 – Esquematização da função ν

$L \backslash M$	0	1	2	3	$\cdots$
	↙	↙	↙	↙	
1	0	1	3	6	
	↙	↙	↙	↙	
2	2	4	7	11	
	↙	↙	↙	↙	
3	5	8	12	17	
	↙	↙	↙	↙	
4	9	13	18	24	
	↙	↙	↙		
$\vdots$					

Fonte: Autor (2024)

Ela está disposta de modo a entendermos a distribuição de ν para contemplar

todo o conjunto dos naturais. Iniciamos olhando para o valor 0 e seguimos as diagonais formadas pelas flechas de cima para baixo. Após uma diagonal, a próxima é a que se inicia, em cima, logo à direita de onde a última diagonal se iniciou.

Como se dará o que faremos? Ao recebermos uma cadeia de entrada, ela pode ser vazia ou não. Se vazia, será equivalente à cadeia vazia da máquina $\mathcal{M}$. Se não, aplicaremos o seguinte.

Consideramos $\{\alpha_1, \alpha_2, \ldots, \alpha_{m'}\}$ o alfabeto de entrada de $\mathcal{M}$. Se a nossa entrada em $\mathcal{M}^{[2]}$ possui exatamente $(x+1)$ símbolos, quer dizer que ela corresponde ao natural x.

Aplicando ν^{-1} a este número, encontramos um par ordenado (k, M_1). O primeiro número deste par indicará a quantidade de símbolos que a cadeia de entrada de $\mathcal{M}$ teria. Se $k = 1$ e $M_1 < m'$, quer dizer que a entrada original é unitária e que seu símbolo é α_{M_1+1}. E se $k = 1$ e $M_1 \geq m'$, rejeitamos tal entrada, pois ela não corresponde a qualquer entrada de $\mathcal{M}$.

Seja qual for o número k, imprimiremos então uma cadeia de $(k+1)$ símbolos 1 no início da fita. Ao lado dela, separadas por $l-1$ espaços em branco, a cadeia de (M_1+1) símbolos 1.

Agora, se $k > 1$, quer dizer que a nossa cadeia de entrada em $\mathcal{M}$ possui exatamente k símbolos. Desta maneira, procederemos como adiante, por $(k-1)$ vezes seguidas.

Ao número M_1 aplicaremos novamente a função ν^{-1}. A partir de M_1, encontraremos o par (m_1, M_2). No lugar da cadeia de (M_1+1) símbolos 1, desde onde ela começava, a substituiremos por uma cadeia de (m_1+1) símbolos 1, com $(l-1)$ quadrados vazios à sua direita, e, por fim, a cadeia de (M_2+1) símbolos 1.

Se $m_1 \geq m'$, rejeitamos a entrada. Se $m_1 < m'$, isso significa que nosso primeiro símbolo da entrada original era α_{m_1+1}. Se $k = 2$, e $M_2 \geq m'$, rejeitamos a entrada. E se $k = 2$ e $M_2 < m'$, então teremos que $M_2 = m_2$, onde o segundo e último símbolo da entrada é α_{m_2+1}.

Agora, se $k > 2$, aplicamos ν^{-1} a M_2, obtendo o par (m_2, M_3), e damos o mesmo seguimento. Ao final, onde não contamos mais a primeira cadeia, dos $(k+1)$ símbolos 1, teremos k cadeias de símbolos 1 em nossa fita, separadas por espaços em branco.

O próximo passo de nossa máquina será reconhecer cada cadeia destas para substituí-la pela sequência binária correspondente ao seu símbolo do alfabeto de entrada. Ao mesmo tempo em que isto é feito, serão removidos os espaços em branco excessivos que separam as sequências. Isto é, aproximaremos as sequências uma da outra. A partir daí, a máquina já inicia a sua computação em réplica à máquina original.

Agora mostraremos que isto é possível com uma máquina de Turing. Quanto ao reconhecimento das cadeias ao final e sua substituição pela versão binária, isto

é mais evidente. Começando pela última das cadeias, fazendo-o da direita para a esquerda, vemos que conseguimos remover os espaços excessivos.

Afinal, teremos quadrados suficientes para, tanto na última cadeia quanto nas outras, substitui-la por uma sequência binária de l quadrados, sem comprometer as demais. Substituímos a última, e então a vamos mover para a esquerda, uma unidade por vez, até nos depararmos com o símbolo 1 mais à direita da cadeia anterior.

A partir dele, paramos, e substituímos agora a cadeia anterior pela versão em sequência binária, e fazemos de novo a mesma coisa, exceto por um passo. Em vez de moveremos apenas a sequência atual para a esquerda, moveremos tanto a sequência atual quanto a última da fita. E continuamos assim até chegar à primeira. E colocamos a primeira, seguida de todas as outras à sua frente, movendo-as para a esquerda para que possamos encaixar nossa cadeia total, de modo que ela comece no início da fita.

Agora, verificaremos que certos passos, anteriores a este que acabamos de empreender, são possíveis com nossa máquina. Tratam-se dos passos relativos aos números triangulares, onde conseguimos os pares ordenados a partir das cadeias de 1. Ao receber a entrada de $(x + 1)$ símbolos 1, nossa máquina de Turing fará uma cópia da entrada para depois de dois quadrados à direita do fim da fita, mas espaçando os símbolos 1 por um quadrado vazio. Assim, temos símbolos 1 de dois em dois quadrados replicando a cadeia do início de fita.

Ela verificará então se o número de símbolos 1 daquela entrada é sucessor de um número triangular. Verá se a cadeia possui exatamente $(1 + 1 + 2 + \cdots + y)$ símbolos 1, para algum y natural positivo. Para isso, ela, da esquerda para a direita, apaga o primeiro 1 copiado e se dirige ao segundo.

Ela marca este segundo 1 copiado com outro 1 e verifica se à direita deste marcador há outro 1. Se há, ela apaga o 1 que acabou de marcar. Depois disso, ela marca o próximo 1 presente de dois em dois quadrados (os 1 da cópia da entrada) com 1 e verifica se à direita deste marcador há outro 1. Se há, ela apaga o 1 recém--marcado. Retorna ao primeiro dos marcadores, o apaga, e se dirige para o próximo 1 de dois em dois quadrados (da cópia) não marcado, se assim houver, para marcá-lo com 1. Verifica se há um 1 à direita do marcador e, se há, apaga o novo 1 marcado.

Ou seja, apagamos o primeiro dos 1 copiados. Marcamos o segundo. Apagamos o segundo e seu marcador e marcamos o terceiro e quarto. Depois apagaríamos o terceiro e quatro e os marcadores, para marcar o quinto, sexto e sétimo. Isto é, verificamos que há $(T_1 + 1)$ símbolos na cadeia de entrada. Depois que há $(T_2 + 1)$. Depois que há $(T_3 + 1)$. E assim por diante.

Se houver outros 1 de dois em quadrados o ciclo continua, se não ele para e verifica quantos marcadores têm até então. Se há M_1 marcadores, é porque o par ordenado é $(1, M_1)$ e, com base neles, colocamos no início da fita uma cadeia de 2

símbolos 1, $(l-1)$ quadrados vazios e uma cadeia de (M_1+1) símbolos 1.

Agora, se foi durante o procedimento de imprimir marcadores que foi verificado que não há um novo 1 de dois em dois, a ação é outra. Mesmo checando a ausência destes respectivos símbolos 1, a máquina completará o serviço de, para cada marcador passado, apagá-lo e deixar um marcador nos quadrados que seriam os intercalantes à frente, embora o faça em estados distintos do quais fazia anteriormente.

Na primeira vez em que verificou que à direita do marcador já não havia 1, não apagou o último 1 marcado. Fez isto para que possa reimprimir os antigos símbolos 1 marcados na última leva de marcadores.

Por exemplo, se agora temos cinco marcadores fechando um ciclo, mas o terceiro destes marcadores ainda possui o 1 à sua esquerda, quer dizer que não havia outro 1 de dois em dois quadrados após este, e que vamos reimprimir os 1 primeiro e segundo marcados. Neste caso, temos $[(T_5-2)+1]$ símbolos 1 na entrada, isto é $(T_5-1-1)+1=\nu(4,1)+1$ símbolos 1.

Diante disso, vemos que, quando esbarramos no meio do ciclo, fecharemos o ciclo com $(k+M_1)$ marcadores e com $(k+1)$ símbolos 1 marcados. Sabendo disso, a máquina coloca, após a cadeia inicial de $(x+1)$ símbolos 1 do início da fita, espaçado dela, uma cadeia de $(k+1)$ símbolos 1, $(l-1)$ quadrados vazios e uma cadeia de (M_1+1) símbolos 1.

O que acabamos de mostrar foi um procedimento, em máquina de Turing, para obter a imagem através da função ν^{-1}. E, por último, há o caso em que, após apagar o primeiro quadrado dos copiados, já não há outros símbolos 1. Quer dizer que a entrada era unitária e, para este caso, o par ordenado é $(1,0)$. A máquina coloca, após a cadeia de $(x+1)$ símbolos 1 do início da fita, uma cadeia de dois símbolos 1, $(l-1)$ quadrados vazios e uma cadeia unitária. E fará o trabalho como a seguir é descrito para qualquer caso com $k=1$.

Depois de feita esta primeira vez, a máquina verificará se a cadeia de $(k+1)$ símbolos é tal que $k=1$ ou não. Se sim, ela apagará a cadeia de $(x+1)$ símbolos do início da fita, e a própria cadeia de $(k+1)$ também, e verifica se na última cadeia $M_1<m'$ ou não. Se é, ele passa para o procedimento de substituição pela sequência binária. Se não é, rejeita.

Agora, se $k>1$ tomaremos a cadeia de (M_1+1) símbolos 1 ao fim da fita e recomeçaremos, a partir dela, o procedimento para encontrar o próximo par ordenado. Mas haverá quatro diferenças.

Primeiro, que agora, para cada vez que for aplicada ν^{-1}, apagaremos o 1 mais à direita da cadeia dos $(k+1)$ símbolos, sempre checando, ainda, um dado. Quando à esquerda do símbolo apagado ainda houver um símbolo 1, nós iremos continuar com este passo. Mas quando não houver, este passo de aplicar ν^{-1} acabará imediatamente, conosco apagando inclusive a cadeia inicial da fita (dos $(x+1)$ símbolos).

Segundo, que faremos a cópia da última cadeia para nela aplicar ν^{-1}, mas não manteremos, depois, a cadeia original. Terceiro que também submeteremos a primeira cadeia do novo par ordenado à verificação a respeito de ela corresponder a um natural menor do que m' ou não. Se corresponde, seguimos. Se não corresponde, a entrada é rejeitada.

E quarto, que cuidaremos para que haja o espaço de $(l-1)$ quadrados vazios entre a antiga penúltima cadeia de símbolos 1 da fita e a primeira cadeia de nosso par ordenado.

Com tudo isto esclarecido, podemos ver que a nossa tarefa consegue ser feita. Deixamos, daí, para em outra seção mais à frente explicar como se daria o restante da computação de uma máquina como $\mathcal{M}^{[2]}$. Em geral, o que faremos segue a mesma ideia. Apenas distinguimos o símbolo 0 do símbolo $\lrcorner$, e adicionamos outro símbolo para separar as sequências binárias.

Já nesta primeira tarefa de distinguir as entradas de nossa nova máquina fica visível que utilizaremos uma grande quantidade de estados. E ainda, quando estivermos trabalhando com as sequência binárias já bem dispostas, para cada estado da máquina original utilizamos basicamente um conjunto de estados para cumprir a tarefa correspondente. Certamente, ao diminuirmos os símbolos, aumentamos em larga escala os estados a serem utilizados, tal como o contrário também é verdade.

E, claro, concluindo esta parte, chegamos à afirmação preparada desde o título desta seção. Existindo, para cada máquina de Turing $\mathcal{M}$, esta versão $\mathcal{M}^{[2]}$ que replica seu trabalho com apenas dois símbolos, fica claro que existe também a máquina $\mathcal{U}^{[2]}$, que replica, nessas condições, o trabalho da máquina universal.

7 REDUTIBILIDADE E COMPUTABILIDADE

Neste momento, daremos atenção a uma relação específica entre linguagens, a partir da qual conseguiremos concluir o status em que se encontram resoluções dos problemas a elas relativos.

7.1 REDUZINDO PROBLEMAS A OUTROS

Retornando algumas páginas, no último resultado da seção 5.4 não precisamos provar diretamente que o *halting problem* era indecidível. Nós mostramos que bastava saber que o *accepting problem* era indecidível, para daí saber que o *halting problem* também o era. E aí, tendo já provado que o *accepting problem* é indecidível, resolvemos o nosso problema.

Isto é, reduzimos o problema da vez a um problema já solucionado anteriormente. Faremos isto outras vezes ainda aqui. Foi muito importante que provássemos a indecidibilidade da linguagem A_{MT}, pois poderemos provar a mesma propriedade para outras linguagens, reduzindo a questão de volta à indecidibilidade de A_{MT}.

Tomemos um alfabeto Σ com que consigamos codificar os objetos desta seção. Eis aqui o problema da vacuidade das máquinas de Turing, elaborado a partir da linguagem

$$\mathsf{V}_{\mathsf{MT}} = \{\langle \mathcal{M} \rangle \in \Sigma^*;\ \mathcal{M} \text{ é uma máquina de Turing e } L(\mathcal{M}) = \varnothing\}.$$

Antes de irmos ao resultado envolvendo isto, vejamos um resultado preliminar.

Lema 7.1.1. *Sejam $\mathcal{M}$ uma máquina de Turing qualquer e w uma entrada a partir do alfabeto de entrada desta máquina. Existe uma máquina de Turing que, ao receber a entrada $\langle \mathcal{M}, w \rangle$, consegue modificar a descrição de $\mathcal{M}$, de modo a fazê-la rejeitar todas as entradas, sem contar w, na qual a máquina modificada continuará agindo do mesmo modo que agiria originalmente.*

A prova deste lema foi deixada no apêndice C. Em um nível de descrição mais abstrato, a máquina em questão, a partir de tal entrada, comporá os primeiros estados e transições da versão modificada de $\mathcal{M}$ de modo que eles só não provocam rejeição na máquina se escaneiam um por um, em ordem, os símbolos de w. Isso se dá sempre comparando os estados e transições compostos com os símbolos de w, do primeiro ao último. Denominaremos a máquina assim construída por $\mathcal{M}[w]$.

A próxima composição será tal que o estado, com a cabeça da máquina mo-

dificada tendo ido para a direita após o fim de w, só admita transição sem rejeição se o símbolo escaneado for $\sqcup$, para garantir que a entrada acaba ali. Dado isso, as composições seguintes levam a cabeça de volta ao início da fita.

E as últimas composições serão a partir da descrição original de $\mathcal{M}$. O último dos estados anteriores deve ser sucedido por aquele estado que correspondia ao antigo estado inicial. E este deve ser sucedido pelos estados correspondentes aos outros estados originais da máquina. Daí, a máquina descrita agora fará o que $\mathcal{M}$ faria com a entrada w.

Um comentário a fazermos é que, para conseguirmos descrever a nova máquina, usamos o artifício de imprimir, em um lugar específico da fita, tantos símbolos (iguais) quanto era o tamanho do alfabeto da fita de $\mathcal{M}$. Isso, pois, para poder cobrir todas as transições. Seguidamente vamos comparando quantos símbolos dessa sequência estão marcados para preenchermos as instruções dos estados e suas transições conforme os símbolos escaneados.

Dito isso, na posse do lema, vamos ao teorema.

Teorema 7.1.2. *A linguagem V_{MT} é indecidível.*

Demonstração. Provaremos que, se tal linguagem fosse decidível, então a linguagem A_{MT} também o seria, o que contradiz o teorema 5.4.5. Para fazê-lo, construiremos a máquina hipotética que decide esta última.

Supondo que V_{MT} seja decidível, existe uma máquina $\mathcal{R}$ que decide tal linguagem. Ou seja, $\mathcal{R}$ aceita exatamente as entradas $\langle \mathcal{M} \rangle$ em que $\mathcal{M}$ é uma máquina de Turing e $L(\mathcal{M}) = \varnothing$, e rejeita as outras.

Também, como de conhecimento, a partir de uma máquina de Turing $\mathcal{M}$ qualquer, podemos construir uma máquina que rejeite todas as cadeias exceto uma escolhida, na qual mantém o comportamentento de $\mathcal{M}$. Se essa cadeia é w, tal máquina construída a partir de $\mathcal{M}$ será a máquina $\mathcal{M}[w]$.

A ideia a ser utilizada é a de que, se a máquina $\mathcal{R}$ diz se uma máquina tem linguagem vazia ou não, então ela fará isso com a máquina $\mathcal{M}[w]$. Se a linguagem desta máquina for vazia, significa que $\mathcal{M}[w]$ não aceita w e, consequentemente, $\mathcal{M}$ não aceita w. E se a linguagem dela for não-vazia, significa que $\mathcal{M}[w]$ aceita w (pois é a única cadeia que ela poderia aceitar) e, consequentemente, $\mathcal{M}$ aceita w.

Sendo assim, vamos construir uma máquina $\mathcal{S}$ como descrita abaixo.

$\mathcal{S}$ = **“** Sobre a entrada $\langle \mathcal{M}, w \rangle$, onde $\mathcal{M}$ é uma máquina de Turing e w é uma cadeia:

1. Elabore a descrição da máquina $\mathcal{M}[w]$.
2. Emule a máquina $\mathcal{R}$ sobre a entrada $\langle \mathcal{M}[w] \rangle$.
3. Se $\mathcal{R}$ aceita, então rejeite. Se $\mathcal{R}$ rejeita, então aceite.**”**

Quando $\mathcal{R}$ aceita a entrada $\langle \mathcal{M}[w] \rangle$, isto significa que $\mathcal{M}$ não aceita w, pois $L(\mathcal{M}[w]) = \varnothing$. E isto está provocando rejeição em $\mathcal{S}$. E quando $\mathcal{R}$ rejeita $\langle \mathcal{M}[w] \rangle$, isto significa que $\mathcal{M}$ aceita w, pois $L(\mathcal{M}[w]) \neq \varnothing$. E isto está provocando a aceitação em $\mathcal{S}$.

Sendo assim, $\mathcal{S}$ é um decisor de $\langle \mathcal{M}, w \rangle$ quanto a $\mathcal{M}$ aceitar w, isto é, $\mathcal{S}$ decide A_{MT}. Como isto não pode ser verdadeiro, segue que V_{MT} é indecidível. ■

Sabendo que temos outras linguagens indecidíveis, para provar a indecidibilidade de outras, podemos reduzir o problema a alguma linguagem que não A_{MT}. Como o problema de decidir se duas máquinas de Turing são equivalentes, elaborado a partir da linguagem

$$EQ_{MT} = \{\langle \mathcal{M}_1, \mathcal{M}_2 \rangle \in \Sigma^*;\ \mathcal{M}_1 \text{ e } \mathcal{M}_2 \text{ são máquinas de Turing e } L(\mathcal{M}_1) = L(\mathcal{M}_2)\}.$$

Para resolver este problema, dessa vez o reduziremos ao problema da decidibilidade de V_{MT}.

Teorema 7.1.3. *A linguagem EQ_{MT} é indecidível.*

Demonstração. Mostraremos que, se esta linguagem fosse decidível, então a linguagem V_{MT} também o seria, o que contradiz o teorema 7.1.2. Para fazê-lo, construiremos a máquina hipotética que decide esta última.

Supondo que EQ_{MT} seja decidível, existe uma máquina $\mathcal{R}$ que decide tal linguagem. Ou seja, $\mathcal{R}$ aceita exatamente as entradas $\langle \mathcal{M}_1, \mathcal{M}_2 \rangle$ em que $\mathcal{M}_1$ e $\mathcal{M}_2$ são máquinas de Turing e $L(\mathcal{M}_1) = L(\mathcal{M}_2)$, e rejeita as outras.

Acontece que, se tivermos uma máquina cuja linguagem sabemos que é vazia, então, com a máquina $\mathcal{R}$, podemos comparar tal linguagem a qualquer outra. Se a máquina $\mathcal{R}$ aceita a entrada com estas duas máquinas, significa que elas têm a mesma linguagem, e, consequentemente, a segunda máquina tem linguagem vazia. E se a máquina $\mathcal{R}$ rejeita a entrada com estas duas máquinas, significa que elas têm linguagens diferentes, e, consequentemente, a segunda máquina tem linguagem não-vazia.

E realmente conhecemos máquinas cuja linguagem é vazia. Elas rejeitam todas as entradas. Para isso, basta que seu estado inicial seja o próprio estado de rejeição. Os outros estados e as transições ficam livres para organizarmos. Tomemos, então, uma máquina $\mathcal{M}_0$ que é assim e, portanto, tem linguagem vazia.

Vamos construir uma máquina $\mathcal{S}$ como descrita abaixo:

$\mathcal{S}$ = " Sobre a entrada $\langle \mathcal{M} \rangle$, onde $\mathcal{M}$ é uma máquina de Turing:

1. Emule $\mathcal{R}$ sobre a entrada $\langle \mathcal{M}, \mathcal{M}_0 \rangle$.

2. Se $\mathcal{R}$ aceita, então aceite. Se $\mathcal{R}$ rejeita, então rejeite."

Quando $\mathcal{R}$ aceita a entrada $\langle \mathcal{M}, \mathcal{M}_0 \rangle$, está dizendo que a linguagem de $\mathcal{M}$ é vazia. E isto está provocando a aceitação em $\mathcal{S}$. E quando $\mathcal{R}$ rejeita $\langle \mathcal{M}, \mathcal{M}_0 \rangle$, está dizendo que a linguagem de $\mathcal{M}$ é não-vazia. E isto está provocando a rejeição de $\mathcal{S}$.

Sendo assim, $\mathcal{S}$ é um decisor de $\langle \mathcal{M} \rangle$ quanto a sua vacuidade, isto é, $\mathcal{S}$ decide V_{MT}. Como isto não pode ser verdadeiro, segue que $\mathsf{EQ}_{\mathsf{MT}}$ é indecidível. ■

7.2 REDUTIBILIDADE POR MAPEAMENTO

Temos reduzido um problema a outro recorrentemente, sem um método sistemático. Agora apresentaremos uma maneira de reduzir um problema a outro, por meio do que chamaremos de **funções computáveis**. Elas servirão para converter instâncias de um problema A em instâncias correspondentes de um problema B.

Definição 12. Seja Σ um alfabeto. Uma função $f : \Sigma^* \to \Sigma^*$ é uma função computável se, e somente se existe uma máquina de Turing $\mathcal{M}$ tal que, para cada entrada $w \in \Sigma^*$, termina sua atividade com exatamente a cadeia $f(w)$ em sua fita.

Há diversos exemplos de máquinas que satisfazem tal condição para uma função computável.

Exemplo 11. Podemos pensar facilmente em uma máquina que, para $a, b \in \mathbb{N}$ positivos, recebe a entrada $\langle a, b \rangle$ e, ao final de seu trabalho, imprime a saída $\langle a+b \rangle$. Tais codificações podem ser feitas a partir do alfabeto $\Sigma = \{\mathsf{Y}, \mathsf{Z}\}$, onde $\underbrace{\mathsf{YY}\ldots\mathsf{Y}}_{a \text{ vezes}}\mathsf{Z}\underbrace{\mathsf{YY}\ldots\mathsf{Y}}_{b \text{ vezes}} = \langle a, b \rangle$ e $\langle a \rangle = \underbrace{\mathsf{YY}\ldots\mathsf{Y}}_{a \text{ vezes}}$.

Tomamos a função $f : \Sigma^* \to \Sigma^*$ tal que

$$f(w) = \begin{cases} \varepsilon, & \text{se } w \neq \overbrace{\mathsf{YY}\ldots\mathsf{Y}}^{a \text{ vezes}}\mathsf{Z}\overbrace{\mathsf{YY}\ldots\mathsf{Y}}^{b \text{ vezes}} \text{ para quaisquer } a, b \in \mathbb{N} - \{0\}, \\ \underbrace{\mathsf{YY}\ldots\mathsf{Y}}_{a+b \text{ vezes}}, & \text{se } w = \underbrace{\mathsf{YY}\ldots\mathsf{Y}}_{a \text{ vezes}}\mathsf{Z}\underbrace{\mathsf{YY}\ldots\mathsf{Y}}_{b \text{ vezes}} \text{ para } a, b \in \mathbb{N} - \{0\} \end{cases}.$$

A máquina $\mathcal{M}$ que montamos para fazer esse trabalho vai procurar o primeiro Z. Se ele estiver no primeiro quadrado, ela apagará todos os símbolos posteriores, e voltará ao começo para apagar o Z e finalizar sua atividade.

Se o Z não for o primeiro, ela vai deixar os primeiros símbolos intactos até encontrar o primeiro Z. Se chegar ao fim e não houver Z, ela apaga todos e termina.

Se houver, ela substitui este Z por um Y e verifica o próximo símbolo à direita. Se ele é um ␣ ou um Z, então ela imprime um ␣ no lugar deste e em todos os outros da direita até encontrar o próximo ␣. Depois volta e apaga todos os símbolos até o primeiro, e então finaliza.

Se imediatamente à direita do antigo primeiro Z havia Y, ela o mantém. E assim manterá enquanto estiver encontrando apenas Y indo para a direita. Se à direita de um Y ela encontra ␣, ela simplesmente volta para este último Y, imprime ␣ no lugar do último Y e finaliza. Mas se ela encontra Z, ela apaga toda a fita, e finaliza.

Tal máquina faz exatamente o que queríamos para podermos afirmar que f é uma função computável. Pois ela termina seu trabalho com apenas uma quantidade de $(a+b)$ símbolos Y no início de sua fita se, e somente se a entrada que recebeu era de a Y's, um Z, e b Y's. E termina sempre com a fita vazia nas outras entradas. □

Exemplo 12. Sejam $\Sigma = \{\widehat{␣}, \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{L}, \mathsf{R}, \mathbf{u}, \widehat{\mathbf{u}}, \mathbf{v}, \mathbf{x}, \widehat{\mathbf{x}}, \mathbf{y}, \widehat{\mathbf{y}}, \mathbf{z}, \widehat{\mathbf{z}}, ;, \#\}$, $\mathcal{M}$ uma máquina de Turing qualquer e u alguma cadeia do alfabeto de entrada de $\mathcal{M}$. Então podemos checar que $f : \Sigma^* \to \Sigma^*$ tal que $f(\langle \mathcal{M}, u \rangle) = \langle \mathcal{M}[u] \rangle$ e $f(w) = \mathsf{A}$ se $w \neq \langle \mathcal{M}, u \rangle$, é uma função computável.

Aqui, $\mathcal{M}[u]$ é a máquina de Turing modificada descrita no lema 7.1.1, a qual rejeita todas as entradas que não u, e que age como agiria $\mathcal{M}$ em u. No apêndice C fornecemos uma máquina $\mathcal{N}$ que, a partir de $\mathcal{M}$ e u, constrói $\mathcal{M}[u]$. Só adicionaremos a esta máquina quatro coisas.

- Quando ela verificar que a entrada que recebe é diferente de $\langle \mathcal{M}, u \rangle$, ela deixará a fita vazia, imprimirá um A no início e depois rejeitará.
- Quando a entrada está de acordo, após ela concluir a descrição da máquina $\mathcal{M}[u]$, que está mais para a direita na fita, ela a trará para o início da fita.
- Os espaços vazios entre dois símbolos distintos de ␣ nessa descrição serão substituídos por $\widehat{␣}$.
- Depois de feitas todas estas últimas duas coisas, no caso da entrada "legítima", a máquina aceita.

Com isso, teremos uma máquina que mostra que f é uma função computável, pois termina em A ou em $\langle \mathcal{M}[u] \rangle$ quando deve. □

Dados os exemplos, agora vamos ao conceito que utiliza as funções computáveis para reduzir problemas a outros.

Definição 13. Sejam Σ um alfabeto e A e B linguagens de Σ. Então dizemos que a linguagem A é **redutível por mapeamento** à linguagem B, denotado por $A \leq_{\mathsf{m}} B$, se

e somente se existe uma função computável $f : \Sigma^* \to \Sigma^*$ onde, para todo $w \in \Sigma^*$, temos que

$$w \in A \iff f(w) \in B.$$

Neste caso, a função f é chamada de **redução** de A para B.

Observe que f não precisa ser injetiva, e muito menos uma bijeção.

Se temos uma redução de A para B, isso significa que os elementos de Σ^* que pertencem a A só têm como imagens elementos de B, e que os elementos de B só são imagens de elementos de A. Consequentemente, os elementos de Σ^* que pertencem a $\overline{A}$ só têm como imagens elementos de $\overline{B}$, e os elementos de $\overline{B}$ só são imagens de elementos de $\overline{A}$.

Observação 8. Fica claro, com isso, pela definição de redutibilidade por mapeamento, que $A \leq_{\mathsf{m}} B$ equivale a $\overline{A} \leq_{\mathsf{m}} \overline{B}$.

Agora, vejamos uma proposição que será futuramente utilizada.

Proposição 7.2.1. *Para toda função computável $f : \Sigma^* \to \Sigma^*$ e toda cadeia $w \in \Sigma^*$, é possível encontrar uma máquina que termine sua atividade com $f(w)$ de maneira que o último quadrado escaneado seja o primeiro da fita.*

Demonstração. Junto de f, temos alguma máquina $\mathcal{M}$ cuja entrada é w e cuja saída é $f(w)$. Esta máquina termina sua atividade em um quadrado qualquer. Ela tem um alfabeto da fita $\Gamma \supset \Sigma \cup \{\textvisiblespace\}$.

Tomando os elementos $\beta \in \Sigma$, vamos definir, para cada um deles, um novo símbolo $\dot{\beta}$ correspondente. Como também, para o símbolo $\textvisiblespace$, definimos o novo símbolo $\dot{\textvisiblespace}$ correspondente. Usando as letras bg, como referência a *begin*, definimos, com isso, o conjunto $\Sigma_{\mathsf{bg}} = \{\dot{\alpha}; \alpha \in \Sigma \cup \{\textvisiblespace\}\}$. Construiremos a máquina bg$(\mathcal{M})$ descrita abaixo, com o mesmo alfabeto de entrada Σ que a máquina $\mathcal{M}$, e com alfabeto da fita $\Gamma \cup \Sigma_{\mathsf{bg}}$.

bg$(\mathcal{M})$ = " Sobre a entrada w:

1. Emule a máquina $\mathcal{M}$ sobre w.

2. Quando no início da fita finalmente estiver a cadeia $f(w)$, e a máquina $\mathcal{M}$ fosse parar, faça o seguinte. Ande para a esquerda, imprimindo $\dot{\alpha}$ no lugar de cada α. Quando escanear o primeiro $\dot{\alpha}$, mantenha-o e vá para a direita. Mantenha cada $\dot{\alpha}$ até encontrar o primeiro α. Quando o encontrar, mantenha-o e vá para a esquerda, substituindo cada $\dot{\alpha}$ por α. Quando escanear o primeiro α, mantenha-o e vá para a esquerda. Se $\mathcal{M}$ aceitava, aceite. Se $\mathcal{M}$ rejeitava, rejeite. "

Com este artifício acima, encontramos o primeiro quadrado. Nós substituímos símbolos α por $\dot{\alpha}$ para fazer isto. O momento em que a cabeça escaneia um $\dot{\alpha}$ é o sinal de que ela já está no primeiro quadrado, pois tentou andar para a esquerda e permaneceu onde estava.

Antes de terminarmos a computação, tratamos de deixar a saída como estava em $\mathcal{M}$. Para isso, a máquina mantém cada $\dot{\alpha}$ e a cabeça vai se dirigir para o fim da fita. Quando lá chegar, retornará para o início de modo que trocará cada um dos $\dot{\alpha}$ pelo α respectivo.

Assim, a máquina bg$(\mathcal{M})$ tem a mesma saída $f(w)$, e vai terminar sua atividade no primeiro quadrado da fita. Sabemos disso pois o momento em que ela escanear um α será quando a cabeça tentou andar para a esquerda, mas na verdade não saiu do primeiro quadrado, no qual ela já substituiu o $\dot{\alpha}$ por α. ■

Observação 9. A partir da descrição $\langle\mathcal{M}\rangle$ de uma máquina de Turing, podemos, através também de uma máquina de Turing, compor a descrição $\langle$bg$(\mathcal{M})\rangle$. Para tal, esta segunda máquina verifica a quantidade m' de B's que há em $\langle\mathcal{M}\rangle$ a fim de acrescentar $(m'+1)$ símbolos na descrição da nova máquina, que serão os $\dot{\alpha}$ citados na demonstração acima.

Ela vai colocar esses novos símbolos nas instruções dos antigos estados, que em verdade não mudará nada na atividade. Depois ela substituirá os estados sucessores que fossem os de aceitação e de rejeição por dois novos estados respectivamente. A nova máquina terá um trabalho, na prática, igual a partir de qualquer um destes dois estados. Exceto que, ao fim das sucessões que começaram no primeiro, a máquina aceitará, e ao fim das sucessões do segundo, a máquina rejeitará. Com isso, basta descrevermos como compor os estados da nova máquina partir de um destes dois, e com o outro será mesmo, com exceção do final.

Seja m a quantidade de símbolos do alfabeto da fita de $\mathcal{M}$. Neste estado substituto do antigo estado de aceitação, descrevemos que, com a leitura dos símbolos α da demonstração, haverá a sua substituição pelo símbolo $\dot{\alpha}$, o qual corresponde à representação do mesmo α, mas com mais m símbolos C. Haverá, concomitantemente, o andar para a esquerda, e o seguir no mesmo estado. Quanto aos símbolos que são puramente do alfabeto da fita, fora o $\sqcup$, a descrição fica livre. E quanto aos símbolos $\dot{\alpha}$, a instrução o manterá, o movimento será de andar para a direita, e mudará para um novo estado sucessor.

Na instrução deste último estado, também serão mantidos os $\dot{\alpha}$ ao escaneá-los, haverá o movimento para a direita e o mantimento do próprio estado. Mais uma vez, os símbolos puramente do alfabeto da fita, sem o $\sqcup$, poderão ter qualquer descrição nesta instrução. Já os α, ao serem escaneados, serão mantidos, o movimento será para a esquerda, e o estado sucessor será outro novo estado.

Com este terceiro novo estado, são substituídos os $\dot{\alpha}$ por α, diminuindo, então,

sua representação em m símbolos C. O movimento da máquina será para a esquerda, e a sucessão mantém o mesmo estado. A descrição das instruções com os símbolos escaneados puramente do alfabeto da fita, sem $\sqcup$, são livres. A descrição para os α é de mantê-los, andar para a esquerda e passar para o novo estado de aceitação.

Fosse então n a quantidade dos estados de $\mathcal{M}$. O que faremos é acrescentar novos estados. Primeiramente com D seguido de, respectivamente, $(n+1)$ e de $(n+2)$ símbolos A, associados ao primeiro tipo dos três novos estados antes apresentados. Aqueles associados ao segundo tipo dos novos estados serão os estados representados por D com respectivamente $(n+3)$ e com $(n+4)$ símbolos A. E, para o terceiro tipo dos novos estados, teremos respectivamente os estados representados por D com $(n+5)$ e com $(n+6)$ símbolos A.

O primeiro estado de cada uma dessas três duplas é o que vai conduzir a máquina à aceitação. O segundo é o que vai conduzir a máquina à rejeição. Assim, do primeiro trio, se segue o estado DE, de aceitação. E do segundo trio, o estado DEE, de rejeição.

Temos, então a construção de bg$(\mathcal{M})$ por meio de uma máquina de Turing.

Agora, a partir da definição da relação $\leq_{\mathsf{m}}$, sistematizamos uma resolução de um primeiro problema a partir de um segundo. E o que nos garante isso é o resultado a seguir.

Teorema 7.2.2. *Se $A \leq_m B$ e B é uma linguagem decidível, então A é decidível.*

Demonstração. Por requisito da definição, existe um alfabeto Σ do qual A e B são linguagens. E, pelas próprias definições, existem uma máquina de Turing $\mathcal{M}_1$ que decide B e uma f redução de A para B. Com esta última, sabemos que há uma máquina de Turing $\mathcal{M}_2$ que recebe as entradas $w \in \Sigma^*$ e acaba em $f(w) \in \Sigma^*$. A partir disso, construiremos a máquina $\mathcal{N}$, de alfabeto de entrada Σ, como descrita abaixo.

$\mathcal{N}$ = " Sobre a entrada w:

1. Emule a máquina $\mathcal{M}_2$ sobre w e nos forneça $f(w)$ ao final.

2. Emule a máquina $\mathcal{M}_1$ sobre $f(w)$. Se $\mathcal{M}_1$ aceitar, aceite. Se $\mathcal{M}_1$ rejeitar, rejeite."

Por f ser redução de A para B, ocorre que $w \in A$ se, e somente se $f(w) \in B$. Como $\mathcal{M}_1$ aceita $f(w) \in B$ e rejeita $f(w) \notin B$, segue que $\mathcal{N}$ aceita $w \in A$ e rejeita $w \notin A$. Portanto, $\mathcal{N}$ é um decisor de A. ■

Corolário 7.2.3. *Se $A \leq_m B$ e A é uma linguagem indecidível, então B é indecidível.* ■

E com este último resultado obtemos diversas provas de indecidibilidade. Podemos revisitar problemas antes resolvidos, onde usamos técnicas de redutibilidade, e vejamos brevemente como poderiam ser resolvidos utilizando redutibilidade por mapeamento. Para isso, eis o exemplo abaixo.

Exemplo 13. No teorema 5.4.7 havíamos abordado o *halting problem*, onde demonstramos a indecidibilidade de PARA_{MT} reduzindo o problema a A_{MT}. Chegamos a uma contradição quando mostramos que um decisor da primeira linguagem poderia ser utilizado para se decidir a segunda. Aqui, vamos ver como faríamos pelo método atual.

Precisamos de uma redução por mapeamento de A_{MT} para PARA_{MT}. Isto é, precisamos de uma função computável f onde, para uma máquina de Turing $\mathcal{M}$ e um cadeia sua w, tenhamos $f(\langle \mathcal{M}, w \rangle) = \langle \mathcal{M}', w' \rangle$, de modo que

$$\langle \mathcal{M}, w \rangle \in \text{A}_{\text{MT}} \iff \langle \mathcal{M}', w' \rangle \in \text{PARA}_{\text{MT}}.$$

E para mostrá-la, construímos a máquina de Turing $\mathcal{F}$ como descrita abaixo.

$\mathcal{F}$ = " Sobre a entrada x:

1. Verifique se ela é uma entrada $\langle \mathcal{M}, w \rangle$. Se sim, siga para o próximo passo. Se não, apague toda a fita, e, com a fita vazia, rejeite.
2. Construa a máquina $\mathcal{M}'$ a seguir.

   $\mathcal{M}'$ = " Sobre a entrada y:

   1. Emule $\mathcal{M}$ sobre a entrada y.
   2. Se $\mathcal{M}$ aceita y, então aceite.
   3. Se $\mathcal{M}$ rejeita y, então entre em loop. "
3. Compute a cadeia $\langle \mathcal{M}', w \rangle$ desde o início da fita e então aceite."

O passo inicial desta máquina garante que uma entrada que não é sequer uma codificação de uma máquina de Turing com uma cadeia não vá parar sobre uma entrada pertencente a PARA_{MT}.

O seu passo seguinte lida com uma codificação "legítima". Ele constrói uma máquina $\mathcal{M}'$ que, ao final, será apresentada com a entrada w que acompanhava a máquina $\mathcal{M}$ inicialmente. Cabe falarmos que não é difícil de se pensar em como construir tal máquina em $\mathcal{F}$.

Digamos que $\mathcal{M}$ tenha n estados. Basta que, quando $\mathcal{F}$ estiver lendo as instruções de $\mathcal{M}$, ela retire o estado de rejeição de onde ele se encontrava como estado sucessor. No lugar dele, colocará um estado novo, o $\mathfrak{q}_n$. Este será acrescentado

nas instruções, como também o estado $\mathfrak{q}_{n+1}$. No primeiro deles, a máquina imprime sempre o símbolo $\sqcup$ sobre qualquer escaneamento, vai para a direita e passa para o estado $\mathfrak{q}_{n+1}$. E no estado $\mathfrak{q}_{n+1}$, a máquina imprime o símbolo $\sqcup$ sobre qualquer escaneamento, vai para a esquerda e passa para o estado $\mathfrak{q}_n$.

Isso garante que $\mathcal{M}'$ entrará em loop quando $\mathcal{M}$ rejeitasse. E não interferirá em $\mathcal{M}'$ imitar $\mathcal{M}$ quando ela aceitasse ou entrasse em loop.

Agora, veja que, se a entrada $\langle \mathcal{M}, w \rangle \in \mathsf{A_{MT}}$, então $\langle \mathcal{M}', w \rangle \in \mathsf{PARA_{MT}}$. Pois, neste caso, $\mathcal{M}$ aceita w. Consequentemente, $\mathcal{M}'$ também aceita w, e disto segue que $\mathcal{M}'$ para sobre w. Assim, $\langle \mathcal{M}', w \rangle \in \mathsf{PARA_{MT}}$.

E também veja que, se a entrada $\langle \mathcal{M}, w \rangle \notin \mathsf{A_{MT}}$, então $\langle \mathcal{M}', w \rangle \notin \mathsf{PARA_{MT}}$. Pois, neste caso, $\mathcal{M}$ não aceita w. Consequentemente, $\mathcal{M}'$ também não aceita w. Ela também não vai rejeitar w, pois tal máquina só chega à aceitação ou entra em loop sobre suas entradas. E disto segue que $\mathcal{M}'$ não para sobre w. Assim, $\langle \mathcal{M}', w \rangle \notin \mathsf{PARA_{MT}}$.

Portanto, $\mathcal{F}$ mostra que f satisfaz aquilo que dissemos, e podemos afirmar que $\mathsf{A_{MT}} \leq_{\mathsf{m}} \mathsf{PARA_{MT}}$. E, pelo corolário 7.2.3, como $\mathsf{A_{MT}}$ é indecidível, segue que $\mathsf{PARA_{MT}}$ também é indecidível. □

Pois bem. Uma vez provados estes resultados relativos a linguagens serem ou não decidíveis, podemos facilmente chegar aos resultados semelhantes relativos a linguagens Turing-reconhecíveis.

Teorema 7.2.4. *Se $A \leq_m B$ e B é uma linguagem Turing-reconhecível, então A é uma linguagem Turing-reconhecível.*

Demonstração. A demonstração é análoga à do teorema 7.2.2. Em vez de lidarmos com máquinas que decidissem A e B, o faríamos com máquinas que as reconhecem. ■

E, tomando o contrapositivo do teorema anterior, obtemos o seguinte.

Corolário 7.2.5. *Se $A \leq_m B$ e A não é uma linguagem Turing-reconhecível, então B não é uma linguagem Turing-reconhecível.* ■

Em posse destes resultados, resolvamos algo concernente ao problema de identificar se duas máquinas de Turing possuem a mesma linguagem. Para tal, recordamos a linguagem $\mathsf{EQ_{MT}}$. Antes havíamos mostrado que ela é indecidível. Agora vamos além.

Teorema 7.2.6. *A linguagem $\mathsf{EQ_{MT}}$ não é Turing-reconhecível e também não é co-Turing-reconhecível.*

Demonstração. Primeiro mostraremos que **i)** $\mathsf{EQ_{MT}}$ não é Turing-reconhecível, e depois, que **ii)** $\overline{\mathsf{EQ_{MT}}}$ também não o é.

Para ambos, utilizaremos o conhecimento tomado do teorema 5.4.6, de que a linguagem $\overline{\text{A}_{\text{MT}}}$ não é Turing-reconhecível, e o corolário 7.2.5.

i) Provaremos que $\overline{\text{A}_{\text{MT}}} \leq_{\text{m}} \text{EQ}_{\text{MT}}$. Para tal, segundo a observação 8, basta provarmos que $\text{A}_{\text{MT}} \leq_{\text{m}} \overline{\text{EQ}_{\text{MT}}}$.

Em vista disso, construímos a máquina descrita abaixo, com alfabeto de entrada Σ.

$\mathcal{F}_1 =$ " Sobre a entrada x:

1. Verifique se ela é uma entrada $\langle \mathcal{M}, w \rangle$, onde $\mathcal{M}$ é uma máquina de Turing e w uma cadeia de entrada dela. Se sim, siga para o próximo passo. Se não, construa a máquina $\mathcal{M}_1$ a seguir.

   $\mathcal{M}_1 =$ " Sobre qualquer entrada:

   1. Rejeite. "

   E forneça a cadeia $\langle \mathcal{M}_1, \mathcal{M}_1 \rangle$ como saída e então aceite.

2. Construa as máquinas $\mathcal{M}_1$, antes descrita, e $\mathcal{M}_2$ a seguir.

   $\mathcal{M}_2 =$ " Sobre qualquer entrada:

   1. Apague toda a entrada.
   2. Emule $\mathcal{M}$ sobre w. Se $\mathcal{M}$ aceitar w, então aceite. Se ela rejeitar, então rejeite. "

3. Forneça a cadeia $\langle \mathcal{M}_1, \mathcal{M}_2 \rangle$ como saída e então aceite."

Ora, se a máquina $\mathcal{M}$ aceita w, então a máquina $\mathcal{M}_2$ aceitará todas as entradas que receber. E se $\mathcal{M}$ não aceitar w, então $\mathcal{M}_2$ não aceitará nenhuma das entradas que receber. No primeiro caso, $L(\mathcal{M}_1) \neq L(\mathcal{M}_2)$, e no segundo, $L(\mathcal{M}_1) = L(\mathcal{M}_2)$.

No primeiro caso, $\langle \mathcal{M}_1, \mathcal{M}_2 \rangle \notin \text{EQ}_{\text{MT}}$, e no segundo, $\langle \mathcal{M}_1, \mathcal{M}_2 \rangle \in \text{EQ}_{\text{MT}}$. Lembrando que não é difícil montar uma máquina tal que $L(\mathcal{M}_1) = \varnothing$. Basta que seu estado inicial seja o estado de rejeição.

E claro, quando $x \neq \langle \mathcal{M}, w \rangle$, a saída da máquina é $\langle \mathcal{M}_1, \mathcal{M}_1 \rangle \in \text{EQ}_{\text{MT}}$. Assim, definindo a função $f_1 : \Sigma^* \to \Sigma^*$ tal que $f_1(x)$ é a saída da máquina $\mathcal{F}_1$ sobre a entrada x, temos uma função computável.

Deste modo, podemos dizer que $x = \langle \mathcal{M}, w \rangle$, onde $\mathcal{M}$ aceita w se, e somente se $f_1(x) \in \overline{\text{EQ}_{\text{MT}}}$. Isto é, $x \in \text{A}_{\text{MT}} \iff f_1(x) \in \overline{\text{EQ}_{\text{MT}}}$. Portanto, $\text{A}_{\text{MT}} \leq_{\text{m}} \overline{\text{EQ}_{\text{MT}}}$.

ii) Provaremos que $\overline{\text{A}_{\text{MT}}} \leq_{\text{m}} \overline{\text{EQ}_{\text{MT}}}$. Para tal, segundo a observação 8, basta provarmos que $\text{A}_{\text{MT}} \leq_{\text{m}} \text{EQ}_{\text{MT}}$.

Em vista disso, construímos a máquina descrita abaixo, com alfabeto de entrada Σ.

$\mathcal{F}_2 =$ " Sobre a entrada x:

1. Verifique se ela é uma entrada $\langle \mathcal{M}, w \rangle$, onde $\mathcal{M}$ é uma máquina de Turing e w uma cadeia de entrada dela. Se sim, siga para o próximo passo. Se não, apague tudo da fita e então rejeite.
2. Construa as máquinas $\mathcal{M}_3$ e $\mathcal{M}_2$ a seguir.

   $\mathcal{M}_3 =$ " Sobre qualquer entrada:

   1. Aceite. "

   $\mathcal{M}_2 =$ " Sobre qualquer entrada:

   1. Apague toda a entrada.
   2. Emule $\mathcal{M}$ sobre w. Se $\mathcal{M}$ aceitar w, então aceite. Se ela rejeitar, então rejeite. "
3. Forneça a cadeia $\langle \mathcal{M}_3, \mathcal{M}_2 \rangle$ como saída e então aceite."

Vendo que o caso de $\mathcal{M}_2$ é o mesmo do item anterior, agora ocorre que, se $\mathcal{M}$ aceita w, então $L(\mathcal{M}_3) = L(\mathcal{M}_2)$, e se $\mathcal{M}$ não aceita w, então $L(\mathcal{M}_3) \neq L(\mathcal{M}_2)$.

No primeiro caso, $\langle \mathcal{M}_3, \mathcal{M}_2 \rangle \in \text{EQ}_{\text{MT}}$, e no segundo, $\langle \mathcal{M}_3, \mathcal{M}_2 \rangle \notin \text{EQ}_{\text{MT}}$. E claro, é fácil montar uma máquina tal que $L(\mathcal{M}_3) = \Sigma^*$. Basta que seu estado inicial seja o estado de aceitação.

E ainda, quando $x \neq \langle \mathcal{M}, w \rangle$, a saída da máquina é $\varepsilon \notin \text{EQ}_{\text{MT}}$. Assim, definindo a função $f_2 : \Sigma^* \to \Sigma^*$ tal que $f_2(x)$ é a saída da máquina $\mathcal{F}_2$ sobre a entrada x, temos uma função computável.

Deste modo, podemos dizer que $x = \langle \mathcal{M}, w \rangle$, onde $\mathcal{M}$ aceita w se, e somente se $f_2(x) \in \text{EQ}_{\text{MT}}$. Isto é, $x \in \text{A}_{\text{MT}} \iff f_2(x) \in \text{EQ}_{\text{MT}}$. Portanto, $\text{A}_{\text{MT}} \leq_{\text{m}} \text{EQ}_{\text{MT}}$. ■

Antes da próxima seção, demonstraremos que há uma importante propriedade dentro da relação de redução por mapeamento.

Proposição 7.2.7. *A relação $\leq_m$ é transitiva.*

Demonstração. Sejam Σ um alfabeto e A, B, C linguagens quaisquer de Σ tais que $A \leq_{\mathsf{m}} B$ e $B \leq_{\mathsf{m}} C$. Quer dizer que existem funções computáveis $f_1, f_2 : \Sigma^* \to \Sigma^*$ tais que $x \in A \iff f_1(x) \in B$ e $y \in B \iff f_2(y) \in C$.

Também quer dizer que existem máquinas de Turing $\mathcal{M}_1$ e $\mathcal{M}_2$ tais que $\mathcal{M}_1$ sobre a entrada x para em $f_1(x)$, e $\mathcal{M}_2$ sobre a entrada y para em $f_2(y)$.

Tomemos a função $f = f_2 \circ f_1$. Ora, $x \in A$ se, somente se $f_1(x) \in B$. E $f_1(x) \in B$ se, e somente se $f_2(f_1(x)) \in C$, isto é, se $f(x) \in C$. Portanto $x \in A \iff f(x) \in C$.

E a função f é computável. Para vê-lo, façamos o seguinte. A partir da máquina $\mathcal{M}_1$, tomemos a máquina bg$(\mathcal{M}_1)$, como definida na demonstração da proposição 7.2.1.

Construiremos a $\mathcal{M}$ como descrita mais abaixo, com mesmo alfabeto de entrada Σ que as máquinas anteriores.

$\mathcal{M} =$ " Sobre a entrada x:

1. Emule bg$(\mathcal{M}_1)$ sobre x.
2. Quando bg$(\mathcal{M}_1)$ fosse parar, sobre a saída $f_1(w)$, emule a máquina $\mathcal{M}_2$ e nos forneça $f_2(f_1(x))$ ao final."

A máquina bg$(\mathcal{M}_1)$ fornece a mesma entrada $f_1(w)$, mas termina no primeiro quadrado da fita. Com isso, podemos prosseguir para emular $\mathcal{M}_2$ sobre $f_1(w)$.

Sendo assim, a máquina $\mathcal{M}$ nos fornece o que precisávamos. A função f é então computável. Logo, $\leq_{\mathsf{m}}$ é transitiva. ■

E ainda, inspirados nesta última demonstração, cabe colocarmos duas coisas: uma definição e uma pertinente observação.

Definição 14. Sejam dadas duas máquinas de Turing $\mathcal{M}_1$ e $\mathcal{M}_2$, cujo alfabeto da fita da primeira contém o da segunda, que sempre exibem saídas e cujas saídas de $\mathcal{M}_1$ são possíveis entradas de $\mathcal{M}_2$. Nós **combinamos** $\mathcal{M}_1$ e $\mathcal{M}_2$ quando, em uma nova máquina $\mathcal{M}$, a entrada w é recebida pela subrotina $\mathcal{M}_1$ e, quando a máquina $\mathcal{M}_1$ chegaria em um estado de parada, $\mathcal{M}$ entra na subrotina $\mathcal{M}_2$, obtendo, da computação de tal máquina, a sua saída sobre w.

Percebamos que, ainda que funções computáveis f e g estejam atreladas às máquinas $\mathcal{M}_1$ e $\mathcal{M}_2$, respectivamente, a saída da máquina que combina as duas sobre uma entrada w não precisa ser $g(f(w))$. Isto porque ao final da primeira subrotina, quando chegar no ponto em que $\mathcal{M}_1$ pararia, pode ser que a máquina não esteja sobre o primeiro quadrado da fita. E é fácil ver que, não estando ali, quando começar a atividade da segunda subrotina, ela agirá segundo os estados de $\mathcal{M}_2$, mas por

começar em outro quadrado, a sucessão de configurações pode mudar radicalmente a saída.

A saída será sempre $g(f(w))$ quando estivermos combinando duas máquinas de Turing das quais a primeira sempre termina no primeiro quadrado da fita. É o caso da máquina resultado da combinação de uma bg $(\mathcal{M}_1)$ com uma $\mathcal{M}_2$.

E agora, com a definição fornecida, vamos à observação e depois passemos para a próxima seção.

Observação 10. Sejam $\mathcal{M}_1$ e $\mathcal{M}_2$ duas máquinas passíveis de combinação. Então existe uma máquina de Turing que, a partir da entrada $\langle \mathcal{M}_1, \mathcal{M}_2 \rangle$, dá como saída a descrição de uma máquina fruto da combinação de $\mathcal{M}_1$ com $\mathcal{M}_2$.

De fato, chamemos tal máquina de $\mathcal{C}$ e mostremos como ela funciona. Sendo n_1 o número de estados de $\mathcal{M}_1$ e n_2 o número de estados de $\mathcal{M}_2$, à descrição dos estados de $\mathcal{M}_1$ é integrada a descrição de $\mathcal{M}_2$.

Adotemos, para explicar que isso é possível, a codificação de máquinas de Turing usual desta dissertação até então, com símbolos, D, A, C, B, etc. Para integrar as duas máquinas, às representações dos estados de $\mathcal{M}_2$ adicionaremos n_1 símbolos A, visto que serão os próximos estados da máquina total.

E sejam m_1 e m_2 as quantidades de símbolos dos alfabetos da fita de $\mathcal{M}_1$ e de $\mathcal{M}_2$, respectivamente. Também adicionaremos os $(m_1 - m_2)$ símbolos (que a primeira máquina tinha e a segunda não) para a descrição de $\mathcal{M}_2$. Na realidade podemos colocar quaisquer instruções em tais configurações, pois a subrotina $\mathcal{M}_2$ nunca trabalhará com estes símbolos.

Se os antigos estados de aceitação e de rejeição da máquina $\mathcal{M}_2$ ficavam respectivamente na posição r_1 e r_2, de 1 a $n_1 - 1$, então agora eles, em vez de serem representados por DE e DEE, o serão por D com $(r_1 + n_1)$ símbolos A e por D com $(r_2 + n_1)$ símbolos A. Estes dois estados serão supérfluos, e já veremos o porquê.

Mas se ou r_1 ou r_2 é a posição 0, então simplesmente apagaremos a descrição de $\mathcal{M}_2$ e daremos como saída a própria $\langle \mathcal{M}_1 \rangle$.

E também, sendo respectivamente r_3 e r_4 as posições do estado de aceitação e do estado de rejeição de $\mathcal{M}_1$, se ou uma ou outra é a posição 0, então apagaremos a descrição de $\mathcal{M}_1$ e daremos como saída a própria $\langle \mathcal{M}_2 \rangle$.

Agora, no caso em que nenhum estado de parada das máquinas originais fosse seu estado inicial, faremos o seguinte. Quanto aos estados de $\mathcal{M}_1$ cujo estado sucessor era o estado de aceitação, agora serão sucedidos pelo estado da posição n_1, o antigo estado inicial de $\mathcal{M}_2$. E fazemos o mesmo com relação ao estado de rejeição quando sucessor na máquina $\mathcal{M}_2$.

Mantemos as instruções dos estados de aceitação e de rejeição da antiga $\mathcal{M}_1$ pois serão os estados de parada da nova máquina. Quando a parte correspondente à antiga $\mathcal{M}_2$ se dirigisse para os estados de parada, é para esses que ela irá. Não irá

nunca para os estados de posição $r_1 + n_1$ e $r_2 + n_1$.

E a quantidade de símbolos B de $\mathcal{M}_1$ será a mesma quantidade usada para os B da nova máquina. Pois o alfabeto de entrada será o mesmo. A combinação está feita.

No entanto, a máquina $\mathcal{C}$ não analisará, quando receber como entrada a descrição de um par de máquinas de Turing, se elas satisfazem as condições para serem combinadas. Ela apenas realizará o seu trabalho.

Dizemos, quando duas máquinas não passíveis de combinação estão como entrada de $\mathcal{C}$, que ela fará uma **tentativa de combinação**. $\mathcal{C}$ vai agir em vista dos passos de adaptar os símbolos nas instruções da segunda máquina e os estados nas duas.

Mas isso pode não resultar em uma combinação. Como quando as saídas da primeira máquina admitissem cadeias que não correspondam a entradas da segunda. Ou quando a primeira máquina entrar em loop para determinadas cadeias de entrada. Ou pode simplesmente não conseguir construir uma máquina de Turing de fato. Como quando o alfabeto da fita da segunda máquina possuía mais símbolos do que o da primeira.

Enfim, ela pode falhar para certos pares de máquinas. Ainda assim, utilizaremos a sua computação em momentos convenientes mais à frente, onde ela não falhará ou onde sua falha não possui relevância.

8 EXPLORANDO A COMPUTABILIDADE

Aqui, faremos uso de um conceito similar para computabilidade, que nos servirá para verificar uma coisa: funções que parecem ser computáveis mas que não o são.

8.1 RADÓ-COMPUTABILIDADE

Poderíamos pensar que é muito complicado encontrar funções de Σ^* em Σ^* que não sejam computáveis. Que precisaríamos de um alfabeto de entrada relativamente grande para chegar ao conhecimento de alguma mais facilmente. Com base em Radó (1962), mostraremos que é possível verificar funções não-computáveis de maneira consideravelmente acessível com um alfabeto binário.

Primeiramente, vamos tratar das definições que o autor utiliza. Usando uma máquina de Turing com apenas dois símbolos, $\sqcup$ e 1, ele toma o alfabeto de entrada $\{1\}$. Em verdade, ele fala de 0 em vez de $\sqcup$, mas ambos cumprem o mesmo papel. Vamos adaptar algumas coisas.

Definição 15. Fixamos a bijeção $\eta : \{1\}^* \to \mathbb{N} \cup \{\varepsilon\}$ como aquela em que

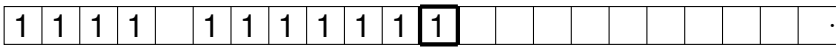
$$\eta(w) = \begin{cases} \varepsilon, & \text{se } w = \varepsilon, \\ x, & \text{se } w = \underbrace{111\ldots11}_{(x+1) \text{ vezes}}. \end{cases}$$

Ele identifica as cadeias não-vazias de $\{1\}^*$ com os números naturais. O número $x \in \mathbb{N}$ é identificado com a cadeia de exatamente $(x + 1)$ símbolos 1 seguidos. Em particular, note que $\eta(1) = 0$.

E, quando fala de uma função computável f, ele toma que ela obtém a imagem $f(x)$ quando acontece o seguinte: logo à direita dos $(x + 1)$ primeiros quadrados da fita, preenchidos com 1, o próximo quadrado está vazio, e à direita dele há exatamente $[f(x) + 1]$ quadrados preenchidos com 1 no momento em que a máquina de Turing para. Neste estágio, o quadrado escaneado é o último da fita.

Por exemplo, nos termos dele, se temos uma função computável tal que $f(3) = 6$, então a máquina para quando está do seguinte modo.

Figura 98 – Exemplo de saída para a computabilidade segundo Radó



Fonte: Autor (2024)

Esta definição não cabe como uma saída dentro da nossa definição de função computável, pois no meio da cadeia final há um $\sqcup$, o qual não está no alfabeto de entrada. No entanto, isto é contornável. Podemos, a esta máquina, adicionar estados que coloquem a última sequência de símbolos 1 logo no início da fita, apagando a primeira sequência.

Em vista disto, trazemos uma definição e um teorema.

Definição 16. Dizemos que uma função $f : \mathbb{N} \cup \{\varepsilon\} \to \mathbb{N} \cup \{\varepsilon\}$ é **Radó-computável** quando:

a) $f(w) = \varepsilon$ se, e somente se $w = \varepsilon$;

b) existe uma máquina de Turing com alfabeto de entrada $\{1\}$ e alfabeto da fita $\{1, \sqcup\}$ tal que, para a entrada ε, a saída é ε, para a entrada $w \neq \varepsilon$, a saída é $w \sqcup \eta^{-1}(f(w))$, e cujo último quadrado escaneado é o último quadrado da fita.

Teorema 8.1.1. *Existe uma função f Radó-computável se, e somente se existe uma função computável $h : \{1\}^* \to \{1\}^*$ tal que $h\left(\eta^{-1}(x)\right) = \eta^{-1}(f(x))$.*

Demonstração. Primeiramente, **i)** partiremos da existência de f para provar a existência de h, e depois **ii)** provaremos a recíproca.

i) Para tal, há a estratégia a seguir. Tomando uma máquina que Radó-computa f, a modificamos a partir do momento em que ela entraria no estado de parada. Em vez de cessar sua atividade, fazemos ela recuar, do último quadrado, até encontrar o quadrado vazio que separa as sequências. Nele imprimimos 1, à esquerda dele imprimimos $\sqcup$ e andamos mais uma vez para a esquerda. Se aí há o símbolo 1, dirigimo-nos para o fim da fita, quando há o 1 seguido de $\sqcup$ e apagamos o 1.

Recomeçamos o mesmo procedimento, até que, na tentativa descrita de ir para a esquerda duas vezes seguidas, na segunda não encontremos o 1, mas o $\sqcup$. Pois isto indica que já estamos no início da fita. Neste estágio, há um quadrado vazio logo no início da fita. O que fazemos é imprimir nele o símbolo 1, ir até o fim da fita, apagar o último símbolo 1, e então chegar ao último quadrado para aí parar.

Está feito. Temos a saída igual a exatamente $[f(x) + 1]$ símbolos 1 seguidos. A entrada era $\overbrace{111\ldots11}^{(x+1)\text{ vezes}} = \eta^{-1}(x)$. A saída é, então $h(\eta^{-1}(x)) = \eta^{-1}(f(x))$. O caso com a entrada ε é evidente.

ii) O caso com a entrada ε dispensa comentários. Quanto aos outros casos, a estratégia aqui será fazer uma cópia da entrada para após o fim da fita, e fazer a

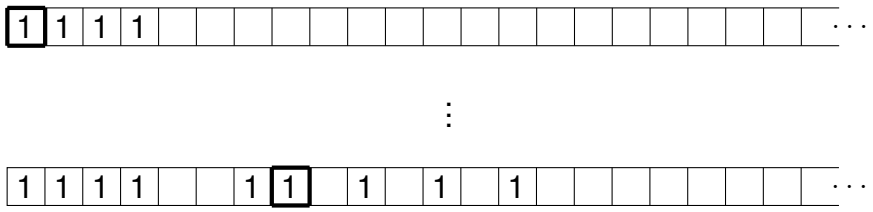
computação acontecer ali. Só que temos de cuidar o caso em que a cabeça da máquina estaria no quadrado correspondente ao primeiro e tentaria ir para a esquerda. Precisamos fazer isto ser imitado sem dano no decorrer da computação.

Faremos o seguinte. Após dois quadrados vazios à direita da entrada, imprimimos 1. Logo à direita dele imprimimos outro 1, pelo primeiro 1 da entrada. Após um quadrado vazio à direita dele, outro 1, pelo segundo 1 da entrada. E vamos imprimindo símbolos 1 de dois em dois quadrados até que, à direita os dois símbolos 1 adjacentes, haja x símbolos 1 intercalados.

Para que consigamos fazer isto, podemos, após imprimir os dois 1 adjacentes, voltar para o fim da entrada original, apagar o último 1 e checar se à esquerda dele há outro 1. Se há, depois disso, retornar ao fim da fita para, imprimir, após um quadrado vazio, outro 1. E ir repetindo este ciclo.

Quando, após apagar um símbolo 1 da fita e à esquerda dele não tiver nada, é porque chegamos ao primeiro quadrado e já teremos imprimido todos os 1 intercalados necessários. Agora, imprimimos de volta o início da fita com símbolos 1 até refazer a entrada original. Paramos de imprimir quando, à direita, já houver outro 1, que se trata do primeiro daqueles adjacentes. Ao encontrá-lo, apagamos os dois símbolos 1 à esquerda dele, e depois andamos à direita até chegarmos ao segundo dos dois adjacentes. Está exemplificado abaixo.

Figura 99 – Exemplo de cópia da entrada para a direita



Fonte: Autor (2024)

Para a atividade restante desta máquina, preparamos os estados baseados nos estados que tinha a máquina que computava a função h. Ela vai agir de maneira semelhante a como agiria tal máquina, escaneando nossa cópia como se fosse a entrada.

Mas, pelo fato de tal cópia ter um espaço vazio entre seus símbolos 1, os estados da máquina original, com exceção dos estados de parada, como que serão divididos em dois estados. Um é usado após o escaneamento de um símbolo na posição dos 1 intercalados, com a cabeça já tendo se deslocado para a esquerda ou para a direita, e o outro é para escanear o símbolo da posição referida.

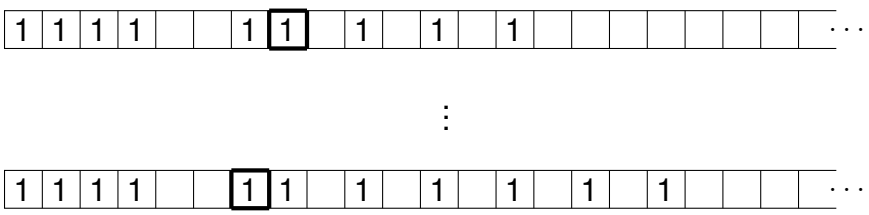
O primeiro destes dois procura ver se a cabeça está sobre um quadrado vazio ou não. Se está, a máquina segue na direção recém-movimentada, e vai para

o segundo estado fruto da divisão, o qual imitará o comportamento da máquina original. Se no entanto, no primeiro dos dois estados, está sobre 1, quer dizer que a cabeça tentou ir para a esquerda do quadrado que corresponde ao primeiro quadrado da fita, e a instrução é que a cabeça retorne para a direita, e continue imitando o comportamento da original.

Com isso, a nova máquina computará de modo semelhante à original, mas, quando antes ocorreria a parada com uma sequência de $\eta(h(x))$ símbolos 1 sobre a fita, agora haverá os $(x+1)$ primeiros 1, seguidos, após dois quadrados vazios, dos dois 1 adjacentes, e, após mais um quadrado vazio, exatamente $\eta(h(x))$ símbolos 1 intercalados. E não será agora a parada. Retornamos ao primeiro quadrado dos dois adjacentes.

Exemplificando, se $\eta(h(3)) = 5$, então podemos checar, abaixo, como seria.

Figura 100 – Exemplo de computação em cima da cópia



Fonte: Autor (2024)

Chegando aí, vamos à sua esquerda imprimir outro 1. Agora a distância entre a entrada original e os adjacentes é de um quadrado. Os últimos passos vêm agora. Temos, atualmente, três adjacentes e, a partir deles, junto dos intercalados, temos $[\eta(h(x)) + 1]$ símbolos 1 até o fim da fita. Iremos juntar $\eta(h(x))$ em uma única sequência contínua no final.

Para isso, à direita dos três adjacentes, andamos até o próximo 1, verificando se dois quadrados à direita dele há outro 1. Se há, voltamos dois quadrados à esquerda, apagamos o 1 nele presente e voltamos à esquerda até encontrar um 1 para, à direita dele, imprimir outro 1. Andamos à direita até o próximo 1 e recomeçamos o ciclo.

Quando, dois quadrados à direita do 1 encontrado no momento em que a cabeça reiniciava o ciclo, não houver outro 1, significa que chegamos ao fim da fita. Voltamos dois quadrados à esquerda, apagamos o 1 ali presente, e voltamos à esquerda até encontrar o último 1 para então o apagar. Está pronto, pois já temos a quantidade exata para a sequência de símbolos 1 após a entrada original, e, chegando no atual último 1 da fita, chegamos ao estado para parar a máquina.

Isto significa que montamos uma máquina que Radó-computa a função f tal que, diante de uma entrada de $(x+1)$ símbolos 1, isto é, a entrada $\eta^{-1}(x)$, obtém como imagem a mesma cadeia que computaria h sobre $\eta^{-1}(x)$. Ora, isto quer dizer que a quantidade de símbolos 1 de $h\left(\eta^{-1}(x)\right)$ é $f(x)+1$. Ou seja, que, $h\left(\eta^{-1}(x)\right) = \eta^{-1}(f(x))$ ■

Isto significa o seguinte. Quando existe uma função Radó-computável f, existe também uma função computável h que, em vez de imprimir a sequência de $[f(x)+1]$ símbolos 1 à direita da cadeia de entrada, a imprime no lugar da própria cadeia de entrada. E vice-versa.

Observação 11. Na realidade, para o autor, diante de uma cadeia de entrada, o primeiro quadrado escaneado será o último quadrado da fita. Entretanto, é evidente que podemos começar pelo primeiro quadrado e imitar o comportamento da máquina segundo ele define. Assim, diante das funções Radó-computáveis f que aparecerem à frente, não distinguiremos a sua capacidade de imprimir $f(x)$ segundo sua definição pede, começando a cabeça da máquina pelo primeiro quadrado da fita ou pelo último.

Originalmente, ele também admite entradas com o símbolo ␣ (ou 0), de modo que, entre uma sequência de símbolos 1 e outra, houvesse um único ␣ que as separasse. Nesta situação, a função f de uma variável vai reconhecer apenas a última sequência de símbolos 1. Se, após um ␣, a última sequência é de $(x+1)$ símbolos 1, a máquina computará $f(x)$, sem se importar com as sequências de 1 anteriores.

Observação 12. Uma coisa importante, que usaremos logo mais, se trata de que, se existe uma função f Radó-computável, e assim, para todo $x \in \mathbb{N}$ existe um $f(x) \in \mathbb{N}$, então existe outra função Radó-computável g tal que, $g(x) = \sum_{i=0}^{x} f(i)$.

Não é tão trabalhoso chegarmos a esta conclusão. Façamos o seguinte. Peguemos a máquina $\mathcal{M}$ que computa f e façamos algumas modificações. Diante da cadeia de $(x+1)$ símbolos 1, ela imprimirá, após o fim da fita, os $[f(x)+1]$ símbolos 1 seguidos. Mas em vez de parar, ela continua.

Ela volta para antes dos últimos $[f(x)+1]$ símbolos 1 e apaga o 1 mais à direita da sequência antecessora. Dirige-se para a esquerda e terá agora x símbolos 1 sequenciados da cabeça para trás. O que fazemos é criar novos estados baseados nos estados originais de $\mathcal{M}$ para que, neste momento, se houver mais símbolos 1 ali, a máquina compute outros $[f(x-1)+1]$ símbolos 1.

Só que estes não serão impressos após um espaço vazio à direita dos x símbolos 1, mas sim seguindo os $[f(x)+1]$ inicialmente impressos, sem espaço vazio que os distancie. Para tal, colocamos estados intermediários entre o reconhecimento da sequência e a impressão dos símbolos após ela, a fim de levarmos a cabeça até o fim da fita carregando a informação devida. E, após isto, apagamos o último 1 da fita, obtendo, até agora, uma sequência $[f(x-1)+f(x)+1]$ símbolos 1 no fim.

Recomeçamos este procedimento, até termos apagado o último (ou primeiro, da esquerda para a direita) 1 da sequência antecessora. Após isto, os reimprimimos, com o único espaço entre a sua sequência e a próxima, que é a última e que já possui exatamente $f(0) + f(1) + \cdots + f(x) + 1 = \left[\sum_{i=0}^{x} f(i)\right] + 1$ símbolos 1. Então, o que dissemos está verificado.

Observação 13. Cabem ainda mais dois rápidos comentários. Com tudo o que percebemos até aqui sobre máquinas de Turing, não é difícil ver que há uma dessas funções Radó-computáveis que consiga realizar o trabalho de dar, como saída de $x \in \mathbb{N}$, a cadeia identificada com $f(x) = x^2$.

E também que, dadas três máquinas relativas a três funções Radó-computáveis f_1, f_2 e f_3 tais que, para $y \in \mathbb{N}$, $f_1(y) \geq f_3(y)$, conseguimos montar uma máquina que Radó-compute g tal que

$$g(x) = \begin{cases} \varepsilon, & \text{se } x = \varepsilon, \\ f_1(x) + f_2(x), & \text{se } x \neq \varepsilon \end{cases},$$

e uma máquina que Rado-compute h tal que

$$h(x) = \begin{cases} \varepsilon, & \text{se } x = \varepsilon, \\ f_1(x) - f_3(x), & \text{se } x \neq \varepsilon \end{cases}.$$

Afinal, fazemos com que as máquinas de g e de h Radó-computem o que seria $f_1(x)$ e, depois, voltando para a entrada, Radó-computem, após o fim da fita, o que seria $f_2(x)$. No caso de g, juntamos as duas cadeias finais em uma só, subtraindo um símbolo 1. No caso de h, apagamos a cadeia de $f_3(x)$ e, para cada símbolo apagado, apagamos um símbolo da cadeia de $f_1(x)$, da direita para a esquerda, acrescentando um símbolo 1 ao final.

Com isso, vamos à próxima seção, cujo conteúdo foi preparado por esta. Conheceremos funções que não são Radó-computáveis, e, conforme o teorema 8.1.1, as funções correlacionadas não-computáveis.

8.2 FUNÇÕES NÃO-COMPUTÁVEIS

Para montar uma máquina com um alfabeto binário tal como aqui apresentamos, com o requisito de não haver estado de parada no estado inicial, pode ser usado o seguinte sistema.

Figura 101 – Exemplo de conjunto de cartões para o *Busy Beaver Game*

C_1				C_2				C_3			
0	1	0	2	0	1	1	1	0	1	1	2
1	1	1	3	1	1	0	2	1	1	0	0

Fonte: Radó (1962)

Representamos os estados da máquina por cartões. O estado inicial será sempre o cartão 1, dito C_1. O segundo cartão, C_2, é outro estado. O terceiro também. As informações que eles carregam são as seguintes:

- Na primeira coluna estão os símbolos a serem escaneados, o 1 ou o 0, que aqui estamos identificando com o ␣.
- Na segunda coluna estão os símbolos a serem impressos neste estado sobre aquele símbolo escaneado da mesma linha.
- Na terceira coluna está representado o movimento da cabeça da máquina, com 0 significando esquerda e 1 significando direita.
- A quarta coluna indica o cartão/estado sucessor.

Neste exemplo, trouxemos três cartões, três estados com os quais uma certa máquina de Turing funciona. Podemos ver que há um estado sucessor indicado por 0, embora não haja um cartão 0. Bom, o cartão C_0 é referente ao estado de parada. Suas instruções, se existissem, seriam supérfluas.

O autor utiliza apenas um estado de parada, sem distinguir entre aceitação e rejeição. Aqui, diremos que, nas máquinas com que estivermos trabalhando, o estado de parada será sempre o estado de aceitação. O estado de rejeição existe, mas nunca será um estado sucessor. Se há n cartões com instruções especificadas, o estado de aceitação será referente ao cartão C_0 e o estado de rejeição será referente ao cartão C_{n+1}.

Com tais cartões, adentramos em um jogo relativo a estas máquinas de Turing: o ***Busy Beaver Game*** (jogo do castor ocupado). Para cada inteiro positivo há uma versão deste jogo, adaptada às regras. Deve-se montar o seu próprio baralho de n cartões, com um inicial, a fim de atingir a pontuação máxima de um certo modo. A seguir, as regras.

1. O jogador, a partir de um número $n \in \mathbb{N} - \{0\}$, escolhe seu baralho ordenado de n cartões, os quais são conforme explicados anteriormente.

2. A máquina montada a partir dos cartões inicia por C_1 em uma fita com a entrada vazia, e deve parar sua atividade após um determinado número s de ***shifts***, isto é, de deslocamentos da cabeça.

3. O jogador inscreve seu baralho e o número s de deslocamentos para a classificação dentro do *International Busy Beaver Club*.

4. O árbitro verifica se a máquina montada com o baralho realmente para em s deslocamentos. É um trabalho relativamente fácil, visto que basta ele reproduzir tal máquina até s deslocamentos. Se ele vê que ela ultrapassa os s, então a inscrição é rejeitada. Se ela para antes dos s, é devolvida ao jogador para que ele faça a correção. E se ela realmente para em s *shifts*, então o ***score***, isto é, a **pontuação** do jogador, será a quantidade de símbolos 1 que a máquina tem na configuração completa final.

Abreviando o nome das versões do jogo para BB-n, o jogador campeão é aquele que obtiver a maior pontuação. O baralho para BB-3 que desenhamos acima, por exemplo, tem pontuação igual a 6. E esta, inclusive, já foi provada ser a maior pontuação possível para BB-3.

Olhando para a regra 4, vemos que uma inscrição válida para o BB-n é um par $(\mathcal{M}, s)$, onde:

a) $\mathcal{M}$ é uma máquina de Turing montada a partir de n cartões especificados;

b) s é um inteiro positivo;

c) $\mathcal{M}$ começa sua atividade pelo cartão C_1 em uma fita com entrada vazia e para após exatos s deslocamentos.

A verificação do árbitro diante das inscrições/pares é um problema decidível. Tratando como linguagem, composta exatamente pelos pares válidos para BB-n, ela é decidível. De fato, basta usar, como subrotina, a máquina universal, fazendo-a emular $\mathcal{M}$ em até s deslocamentos.

Devemos notar que, se duas inscrições $(\mathcal{M}_1, s_1)$ e $(\mathcal{M}_2, s_2)$ são tais que $\mathcal{M}_1 = \mathcal{M}_2$, então, obviamente, $s_1 = s_2$. Visto isso, há um número limitado de máquinas de Turing a serem montadas dentro de BB-n, que podem ou não fazer parte de inscrições válidas.

Para o caso de em um estado a máquina escanear o 0 (ou seja, ␣), há duas possibilidades de símbolo a imprimir (0 ou 1), há duas possibilidades de movimento (esquerda ou direita) e há $(n + 1)$ possibilidades de estado sucessor (de C_0 até C_n), sendo que há n estados a terem tais instruções especificadas, e tudo isto ainda é considerado novamente devido ao potencial escaneamento do 1. Sendo então $N(n)$

o número total das máquinas de Turing montadas a partir dos baralhos de n cartões especificados, temos que

$$N(n) = [4(n+1)]^{2n}.$$

Também é verdade que sempre existe alguma inscrição válida para BB-n. Basta que no cartão C_1, na linha referente ao escaneamento do 0, a instrução indique 1 1 0, onde a máquina imprime 1, anda para a direita e vai para o estado de parada. Teremos uma inscrição $(\mathcal{M}, 1)$ válida para BB-n. Na verdade mais de uma, pois esta carrega consigo várias possibilidades de instruções para a linha referente ao escaneamento do 1 no mesmo cartão, como também várias possibilidades de instruções para os outros $(n-1)$ cartões.

Com isso, seja E_n o conjunto das inscrições válidas para BB-n. Então já sabemos que ele é um conjunto não-vazio. Sendo $N_e(n)$ a quantidade de elementos de E_n, então podemos ver que se trata de um número finito tal que

$$1 < N_e(n) < N(n) = [4(n+1)]^{2n}.$$

E claro, conseguimos sempre decidir se um par $(\mathcal{M}, s)$ é ou não uma inscrição válida. Assim, podemos dizer que o conjunto E_n é bem-definido. Ele é um conjunto não-vazio e finito, então, a princípio, imaginamos que a função N_e, de sua quantidade para cada n, é Radó-computável. Afinal, temos até uma cota superior para ela, dada por uma expressão bastante simples. Mas, surpreendentemente, ela não o é. E veremos isto ao final desta seção.

Antes, veremos o caso de uma função envolvendo as pontuações de cada inscrição válida. Seja, para cada $(\mathcal{M}, s) \in E_n$ (as inscrições válidas), $\sigma(\mathcal{M}, s)$ a pontuação desta inscrição. Olhando para o conjunto das pontuações conseguidas a partir de um E_n, claramente, mais uma vez temos um conjunto bem-definido, não-vazio e finito. É composto todo por inteiros não-negativos.

O que checaremos é que, a ele, está atrelada uma função que também parece Radó-computável, mas que não o é. Trata-se da função $\Sigma_{\mathsf{m}} : \mathbb{N} \cup \{\varepsilon\} \to \mathbb{N} \cup \{\varepsilon\}$ definida por

$$\Sigma_{\mathsf{m}}(\varepsilon) = \varepsilon,$$

$$\Sigma_{\mathsf{m}}(0) = 0,$$

e

$$\Sigma_{\mathsf{m}}(n) = \max\left[\sigma(\mathcal{M}, s)\right], \text{ para } (\mathcal{M}, s) \in E_n.$$

Embora não seja Radó-computável, tomando um valor particular para n é perfeitamente possível encontrarmos o valor de $\Sigma_{\mathsf{m}}(n)$, visto que podemos definir o conjunto E_n, a partir do qual testaríamos todas as possibilidades. Agora, para chegarmos à conclusão de que ela não é Radó-computável, tomemos uma notação que nos au-

xiliará.

Definição 17. Sendo f e g duas funções Radó-computáveis, nós diremos que f **ultrapassa** g se, e somente se existe algum $n_0 \in \mathbb{N}$ de modo que, se $n > n_0$, então $f(n) > g(n)$. Isto será denotado por $f(x) \succ g(x)$.

Dado isso, vamos provar o resultado preliminar ao antes anunciado.

Lema 8.2.1. *Para toda função Radó-computável f, ocorre que $\Sigma_m(x) \succ f(x)$.*

Demonstração. Tomemos uma função f Radó-computável qualquer. Definamos, com ela, a função $F : \mathbb{N} \cup \{\varepsilon\} \to \mathbb{N} \cup \{\varepsilon\}$ tal que $F(\varepsilon) = \varepsilon$ e

$$F(x) = \sum_{i=0}^{x} \left[f(i) + i^2 \right].$$

Resgatando aquilo que dissemos na observação 13, fica claro que a função F é Radó-computável. Como também são claras, para $x \in \mathbb{N}$, as seguintes afirmações:

$$F(x) \geq f(x), \tag{8.1}$$

$$F(x) \geq x^2, \tag{8.2}$$

$$F(x+1) > F(x). \tag{8.3}$$

Logo, existe uma máquina de Turing $\mathcal{M}_F$ que Radó-computa F. E, para cada $x \in \mathbb{N}$, definimos uma máquina de Turing binária $\mathcal{M}^{(x)}$ que, sobre a entrada vazia, imprime $(x+1)$ símbolos 1 e para com a cabeça sobre o último deles.

Tal máquina pode ser construída com exatamente $(x+1)$ cartões especificados. Os x primeiros, na linha de instrução do escaneamento do 0, servem para imprimir os x primeiros símbolos 1, andar para a direita e ir para o próximo cartão da fila. O cartão C_{x+1} imprime o último 1, anda para a esquerda e seu sucessor pode ser qualquer cartão, desde que este, na linha do escaneamento do 1, imprima 1, ande para a direita e cujo sucessor seja C_0. Com isso, está feito.

Como vimos no teorema 8.1.1, sendo F Radó-computável, existe uma função computável h cujas saídas sobre $\eta^{-1}(x)$ são $\eta^{-1}(F(x))$. Então há uma máquina de Turing binária $\mathcal{M}_h$ que computa h. E, relembrando a observação 11, não precisamos nos preocupar se a atividade de $\mathcal{M}_F$ começa no primeiro ou no último quadrado da fita, e o mesmo observamos quando, a partir dela, obtemos $\mathcal{M}_h$. E seja C o número de cartões especificados de $\mathcal{M}_h$.

Sendo assim, façamos o seguinte. Podemos ver que $\mathcal{M}^{(x)}$ e $\mathcal{M}_h$ são passíveis de combinação. Assim, combinemos, respectivamente, as máquinas $\mathcal{M}^{(x)}$ e $\mathcal{M}_h$,

e, depois, combinemos o resultado com outra máquina $\mathcal{M}_h$. Chamemos esta nova máquina de $\mathcal{M}_h^{(x)}$. Ela possui $(1 + x + 2C)$ cartões especificados.

Esta máquina de Turing imprime os $(x+1)$ símbolos 1 na primeira parte. Segue para a segunda, partindo do fim de tal cadeia, para imprimir então $[F(x) + 1]$ símbolos 1 no lugar. E segue para a terceira parte, partindo também do fim, para imprimir, no lugar, $[F(F(x)) + 1]$ símbolos 1. E então para no último quadrado.

Tomando a função da pontuação máxima, e olhando para a pontuação atrelada à máquina $\mathcal{M}_h^{(x)}$, então é verdade que

$$\Sigma_{\mathsf{m}}(1 + x + 2C) \geq F(F(x)). \tag{8.4}$$

Ora, é evidente que $x^2 \succ 1 + x + 2C$, e tendo isto e (8.2) em mente, temos que

$$F(x) \succ 1 + x + 2C. \tag{8.5}$$

Tomando (8.3), vemos que, dentro dos naturais, a função F sempre cresce. Com isso, e com (8.5), segue que

$$F(F(x)) \succ F(1 + x + 2C). \tag{8.6}$$

Juntando as afirmações (8.4) e (8.6), chegamos ao fato de que

$$\Sigma_{\mathsf{m}}(1 + x + 2C) \succ F(1 + x + 2C).$$

Com isto, e com (8.1), temos que

$$\Sigma_{\mathsf{m}}(1 + x + 2C) \succ f(1 + x + 2C).$$

Deste modo, tomando $n_0 = 1 + x + 2C$, para todo $n > n_0$, temos que $\Sigma_{\mathsf{m}}(n) > f(n)$ e, portanto, $\Sigma_{\mathsf{m}}(n) \succ f(n)$. ■

Teorema 8.2.2. *A função Σ_m não é Radó-computável.*

Demonstração. Pelo lema anterior, $\Sigma_{\mathsf{m}}(x) \succ f(x)$ para toda função f Radó-computável. Ora, por definição, não é possível que $\Sigma_{\mathsf{m}}(x) \succ \Sigma_{\mathsf{m}}(x)$. Logo, Σ_{m} não é Radó--computável. ■

Com o teorema acima, e o teorema 8.1.1, segue que não existe uma máquina que, sobre a entrada $\eta^{-1}(x)$ obtenha a saída $\eta^{-1}(\Sigma_{\mathsf{m}}(x))$. Concluímos algo a mais então.

Corolário 8.2.3. *A função $f : \{1\}^* \to \{1\}^*$ tal que $f(\overbrace{111\ldots11}^{(x+1)\ \textit{vezes}}) = \overbrace{111\ldots11}^{[\Sigma_m(x)+1]\ \textit{vezes}}$ não é computável.* ■

Agora, olhemos para outra função, $S : \mathbb{N} \cup \{\varepsilon\} \to \mathbb{N} \cup \{\varepsilon\}$, tal que

$$S(\varepsilon) = \varepsilon,$$

$$S(0) = 0,$$

e

$$S(n) = \max(s), \text{ para } (\mathcal{M}, s) \in E_n.$$

Trata-se da função que, em suma, obtém, a partir de $n \in \mathbb{N} - \{0\}$, o maior número de deslocamentos de uma máquina de n cartões especificados que não entre em loop sobre a entrada vazia. Estas são todas as máquinas de E_n, e, claramente, estão em número finito positivo.

Também é claro que o número de símbolos 1 impressos na saída destas máquinas tem que ser menor ou igual ao seu número de deslocamentos. Deste modo, a maior das pontuações não pode ser maior que o máximo de deslocamentos, ambos para um mesmo n. Assim, é verdade que

$$S(n) \geq \Sigma_{\mathsf{m}}(n)$$

e, consequentemente,

$$S(n) \succ f(n)$$

para toda função f Radó-computável. E segue, daí, o resultado abaixo.

Teorema 8.2.4. *A função S não é Radó-computável.*

Como também a consequência.

Corolário 8.2.5. *A função $f : \{1\}^* \to \{1\}^*$ tal que $f(\overbrace{111\ldots11}^{(x+1)\ \textit{vezes}}) = \overbrace{111\ldots11}^{[S(x)+1]\ \textit{vezes}}$ não é computável.* ■

Observação 14. Esta última informação diz respeito exatamente à indecidibilidade do *halting problem*. Pois se, para um dado número de cartões especificados, por algum algoritmo universal soubéssemos qual é o número máximo de deslocamentos que a cabeça da máquina faz antes de parar, bastaria checarmos para cada uma delas quais não param até percorrerem este número de deslocamentos. Conseguiríamos decidir quais máquinas entram e quais não entram em loop, o que já vimos que é impossível.

É claro, aqui estamos desconsiderando transições para o estado de rejeição. No entanto, é visível que, essencialmente, o problema ainda é o mesmo e que chegamos à mesma conclusão.

E, por fim, demonstraremos a incomputabilidade de mais uma das funções envolvidas no *Busy Beaver Game*.

Teorema 8.2.6. *A função N_e não é Radó-computável.*

Demonstração. Mostraremos que se N_e fosse Radó-computável, então S também o seria. E, pela contrapositiva, concluiremos que N_e, não o é.

Basta vermos que se N_e é Radó-computável, então conseguimos determinar, a partir de uma entrada $\overbrace{111\ldots11}^{(n+1)\text{ vezes}}$, a quantidade de elementos de E_n com uma máquina de Turing. Sabendo disso, após encontrarmos o valor de $N_e(n)$, o utilizaríamos como um delimitador.

Encontrado este valor, o nosso próximo passo com a máquina é reproduzir a computação de todas as máquinas de BB-n (que são finitas) e verificar quais param para diferentes valores de s deslocamentos. Para fazer tal reprodução, ela checa o valor de n que recebeu e vai variando, nas instruções das máquinas de BB-n que escrever, o símbolo a imprimir, a direção da cabeça e o estado sucessor segundo uma ordem. Isso é perfeitamente possível, e ao fim ela cobrirá todas as máquinas de BB-n.

Fazer isto apenas com símbolos 1 e $\sqcup$ leva mais tempo, mas conseguimos fazer acontecer. Basta dispor bem as cadeias de 1 e os espaçamentos entre elas, tal como fizemos outras vezes aqui. Para verificar os valores de s, realizaremos a computação de cada máquina de BB-n com a subrotina de uma máquina universal, tal como aquela de que comentamos na seção 6.2, de apenas dois símbolos.

Consultando as instruções montadas de cada vez, nossa máquina testa cada uma das simulações que faz sobre o que seria uma fita vazia, parando em exatamente s deslocamentos, variando tal valor desde 1 até os seguintes na ordem crescente. Ela vai registrando quais destas máquinas chegaram à aceitação com este número de deslocamentos e vai imprimindo uma marcação cada vez que encontra uma máquina dessas.

Em algum momento ela vai conseguir percorrer todas as máquinas que estão em BB-n. E saberá disso quando encontrar a $[N_e(n)]$-ésima máquina que parar em uma certo valor de s. E este valor de s será, portanto, o máximo possível para BB-n, isto é, será $S(n)$.

Portanto, conseguiríamos registrar tal valor em uma cadeia de $[S(n)+1]$ símbolos 1 ao lado da cadeia inicial de $(n+1)$ símbolos 1 da fita, apagando o restante da fita. Com isso, concluiríamos que a função S é Radó-computável.

Mas sabemos que isto contradiz o teorema 8.2.4. Logo, a função N_e não é Radó-computável. ■

E, obviamente, a consequência.

Corolário 8.2.7. *A função $f : \{1\}^* \to \{1\}^*$ tal que $f(\overbrace{111\ldots11}^{(x+1)\ \textit{vezes}}) = \overbrace{111\ldots11}^{[N_e(x)+1]\ \textit{vezes}}$ não é computável.* ■

Assim, chegamos ao conhecimento de algumas funções relativamente simples, mas que se apresentam como não-computáveis.

9 ÚLTIMAS BASES

Usamos este capítulo para preparar os próximos dois. Gastamos tais páginas para, quando tratarmos das demonstrações finais, já termos em nossa mente as noções claras de alfabetos a serem usados e de como nos referirmos ao histórico de trabalho de uma máquina de Turing a partir de uma entrada.

9.1 ALGUMAS CONVENÇÕES

Aqui retornamos ao que havíamos começado na seção 6.2, com relação a retratar símbolos diversos a partir de sequências binárias. Convém definirmos um conjunto que utilizaremos daqui a umas seções.

Definição 18. Denominaremos Σ_{e} o alfabeto $\{0, 1, @\}$, e Γ_{e} o alfabeto $\Sigma_{\mathsf{e}} \cup \{␣\}$.

A letra e se dá em referência à palavra *every*. A partir deste alfabeto iremos representar codificações de cadeias que antes eram compostas por símbolos de outros alfabetos, e poderemos fazer isso com todos os alfabetos que imaginarmos.

Muito falamos de funções computáveis e trabalhamos com elas. Só que elas sempre são funções de um conjunto sobre ele mesmo, o qual deve ser algum alfabeto sob a operação estrela. Em uma dada situação poderíamos querer considerar, mais naturalmente, a uma função $f : \Sigma_1^* \to \Sigma_2^*$. E, sendo estes dois alfabetos apresentados diferentes, não terminamos em uma função computável.

Agora, fosse Γ o alfabeto da fita da máquina que computaria, a partir de uma entrada x, a saída $f(x)$. E fosse m a quantidade de elementos de Γ. Se Γ é igual a $\{␣\}$, então o que faremos a seguir é dispensável. Agora, se Γ e possui mais de dois símbolos, existe um $K \in \mathbb{N}$ tal que $2^K + 1 < m \leq 2^{K+1} + 1$.

Tendo noção disso, entendemos que somos capazes de identificar cada símbolo de $\Gamma - \{␣\}$ com um número de 0 até $m - 2$ representados na base binária. Mais precisamente, representados com K algarismos binários seguidos.

Para isso utilizamos os símbolos 0 e 1. E utilizamos o símbolo @ para separar entre si as representações de símbolos de Γ. Como o símbolo ␣, por causa da definição de máquina de Turing, tem que estar presente, ele fica representado por ele mesmo. E por isso teremos, para alfabeto da fita, o conjunto Γ_{e}.

Mais precisamente, representaremos o símbolo ␣ de Γ utilizando exatamente K símbolos ␣ seguidos. Isto se dá para que, quando substituído pela sequência referente a outro símbolo de Γ, o espaço a ocupar na fita seja o mesmo.

É claro, por vezes, em uma cadeia de entrada, pode aparecer uma sequência de 0's e 1's, entre dois @'s, que exceda o número $m-2$ na base binária. Isso significa que conseguimos representar todos os símbolos de Γ e também uma quantidade infinita de outros símbolos na mesma máquina.

Como queremos representar somente os símbolos de Γ, trataremos de contornar este problema. Isto pode ser feito de várias formas. Por exemplo, quando a máquina em que estivermos verificar que, entre dois @'s, sem outro @ no meio, houver mais do que $(K+1)$ símbolos 0 e 1, ela considera esta entrada como "ilegítima" e a descartará, fazendo algo como apagar toda a fita e então rejeitar. E, a fim de facilitarmos nossa linguagem daqui para frente, diremos que uma máquina de Turing **descarta** uma entrada quando, logo após verificá-la, ela apaga toda a fita e então rejeita.

Bom, se estávamos codificando um alfabeto cuja quantidade de símbolos era ainda $m \leq 2^{K+1}$, então nos restam $\left(2^{K+1}-m+1\right)$ símbolos que supostamente estão sendo representados na base binária e que não possuem um correspondente em Γ. Para isto devemos ter, em nossa máquina, os comandos específicos para descartar entradas que possuam sequências com números binários de $m-1$ até $2^{K+1}-1$.

Por exemplo, se o alfabeto da fita tivesse 6 símbolos. Então haveria 5 símbolos representados na base binária, desde o 000 até o 100. O que faríamos é deixar nossa máquina de tal modo que rejeitasse entradas que trouxessem as sequências 101, 110 e 111. No restante da computação, após a leitura da entrada, ainda poderia aparecer a sequência $\sqcup\sqcup\sqcup$, que representaria o $\sqcup$ de Γ.

E ainda, há a distinção entre o alfabeto de entrada e o alfabeto da fita. Digamos que fosse m' a quantidade de símbolos do alfabeto de entrada. Para resolver isso, estenderemos o procedimento de agora há pouco. Ao verificar a cadeia de entrada, a máquina a descartará se houver sequências da base binária que excederem a representação de $m'-1$ até o $2^{K+1}-1$.

Assim, se o alfabeto de fita tinha 6 símbolos, mas o alfabeto de entrada tinha apenas 3, então seus elementos serão representados por meio das sequências desde 000 até 010. E, na entrada, a máquina descartará as cadeias que trouxerem alguma sequência dentre as seguintes: 011, 100, 101, 110 e 111. O que não quer dizer que, após isso, ao longo da atividade da máquina, não haverá sequências 011 ou 100. Poderá haver sim, para representar os símbolos do alfabeto da fita original. Como também a sequência $\sqcup\sqcup\sqcup$. Já as sequências restantes, no caso das 101, 110 e 111, nem isso.

Não precisamos nos preocupar com os casos em que o alfabeto da fita teria um ou dois símbolos. Como também no caso em que o alfabeto de entrada fosse vazio. São situações excepcionais e que também possuem representações possíveis com os alfabetos que definimos, embora possam ser dispensáveis.

Ao lado do @, nós usamos o 0 e o 1, mas na realidade poderíamos usar quaisquer dois símbolos distintos para este papel de englobar todos os alfabetos. Porém, convencionamos assim mesmo, é comum.

Ainda, é proveitoso ter uma noção mais apurada de como poderia ser o alfabeto da fita, Γ, formado para satisfazer o que falamos acima. E também como poderia ser o alfabeto de entrada, Σ, formado a partir de alfabetos Σ_1 (aquele sob a operação estrela no domínio de uma função) e Σ_2 (no contradomínio).

É possível termos um Σ igual a $\Sigma_1 \cup \Sigma_2$. Com ele, já conseguimos ter acesso a uma função computável. Pois as entradas e as saídas já possuem símbolos dos alfabetos que queríamos. E podemos descartar cadeias de entrada indesejadas. Por exemplo, cadeias que possuam algum símbolo de Σ_2, fazendo com que, na prática, a nossa máquina conseguisse trabalhar tal como a função de Σ_1^* em Σ_2^*.

E, neste caso, o alfabeto Γ pode ser a mesma união, junto de todos os símbolos que a máquina de Turing utilizasse para computar ao longo de seu trabalho até a saída.

Vendo que existe tal função computável, de Σ^* em Σ^* que cumpre o mesmo papel que uma função de Σ_1^* em Σ_2^*, vamos a um resultado a respeito de convertermos tal máquina para uma máquina de Turing que funcione a partir de Σ_e e Γ_e.

Teorema 9.1.1. *Para cada função computável $f : \Sigma^* \to \Sigma^*$ existe uma função computável $f_e : \Sigma_e^* \to \Sigma_e^*$ tal que, para todo $w \in \Sigma^*$, $f_e(\langle w \rangle) = \langle f(w) \rangle$.*

Demonstração. Seja $\mathcal{M}$ uma máquina que computa f, e Γ o alfabeto da fita desta máquina. Usando o procedimento que vínhamos descrevendo, identificando cada símbolo de Γ com um número na base binária desde o zero, com exceção do $\sqcup$.

Colocamos cada sequência binária entre dois @'s. Descartamos entradas que tiverem sequências binárias que ultrapassem o número de dígitos e entradas que não o ultrapassem mas cujas sequências não estão identificadas com símbolos de Γ. Também o fazemos para entradas com alguma sequência identificada com elementos que pertencem puramente a Γ.

Além disso, se o mínimo de casas para representar todos os elementos de Γ na base binária é K, fixamos que todos os símbolos de Γ serão representados com K quadrados. Para isto, basta acrescentar dígitos 0 no início de algumas das representações. Então, entradas com sequências binárias de menos de K quadrados também serão descartadas. O caso do $\sqcup$ de Γ será semelhante, pois o representaremos com K quadrados vazios seguidos.

Sendo assim, as transições para a nova máquina serão do seguinte modo. O primeiro estado da nova máquina reconhece se o primeiro símbolo é @. Se não é @ e nem $\sqcup$, ocorre o descarte. Se é @, a cabeça vai para a direita e há $(K-1)$ estados sucessores que verificam se há 0 ou 1. Se com algum deles a máquina encontra o $\sqcup$ ou o @, há o descarte. Eles levam sempre a cabeça da máquina para a direita.

Ao último deles sucede-se um estado que visa reconhecer se há um @, estado este distinto do estado inicial. Se não há, descartamos. Se há, seguimos para a direita e repetimos os ciclo, recomeçando pelos $(K-1)$ estados. A máquina só vai considerar a entrada como legítima se, à direita o estado para reconhecer @ que não o inicial, houver um quadrado vazio.

Depois disso a máquina vai para a esquerda em um mesmo estado, até que se encontre com um @. Se no movimento para a esquerda deste símbolo não houver outro @, ela retorna ao estado antecessor, indo para a esquerda. Se houver outro @, significa que ela chegou ao início da fita.

Depois disso, a máquina fará novamente o caminho do início ao fim da fita, mas para reconhecer se as sequências binárias apenas representam símbolos do alfabeto de entrada de $\mathcal{M}$. Se não, descarta. Se sim, volta ao início da fita.

Todos os estados acima descritos reimprimem o símbolo nos quadrados, exceto o estado inicial, que tem um comportamento diferente se ele escaneia $\sqcup$. Mas por enquanto, trabalhamos olhando para o caso em que não foi assim.

O próximo passo é o de escanear toda a primeira sequência binária de K símbolos para reconhecer a qual símbolo de Σ ela se refere. Com o último destes estados encontra o @, já com a informação necessária. Começa a voltar para a esquerda em um mesmo estado até encontrar o @. Ali, muda para o estado a fim de que comecemos a imprimir a sequência que substitui os símbolos naqueles K quadrados, indo para a direita. Nisto, a máquina passa por K estados até chegar no @.

Agora, se o símbolo que o estado inicial escaneou foi $\sqcup$, ele imprime ali o @. A máquina anda para a direita, seguida de $(K-1)$ estados que andam para a direita sem imprimir algo novo. O próximo estado imprime outro @, e o comportamento a seguir é o mesmo descrito no parágrafo anterior, visto que, consequentemente, a informação atual é a de que a primeira sequência corresponde ao símbolo $\sqcup$ em Γ.

Tanto neste caso quanto no caso anterior, largamente descrito, o próximo passo será o descrito nos próximos parágrafos. Mas precisávamos citar o presente caso, pois a entrada vazia também corresponde a uma cadeia $w \in \Sigma^*$.

Daí, começamos a trabalhar a partir dos movimentos e impressões da máquina original. Se vamos nos basear em um estado que levava a cabeça de $\mathcal{M}$ para a direita, aqui passaremos por K estados que levam para a direita desde o @ que finalizava a sequência anterior, os quais reconhecem qual é o símbolo de Γ representado por tal sequência.

Depois vem um estado que chega ao próximo @ ou a um $\sqcup$ (isto se já ultrapassamos o fim da fita). A presença deste estado específico atual já é consequência da informação de qual símbolo de Γ está representado naquela sequência. Com ele, a máquina imprime @ e vai para a esquerda, sendo ele sucedido por um outro estado que se repetirá levando a cabeça para a esquerda até encontrar o @. Quando chega

nele, a máquina muda de estado e vai para a direita, de modo a percorrer mais K estados com os quais imprimirá a nova sequência naquele espaço, correspondente ao símbolo que sucederia aquele representado anteriormente. O próximo estado chegará no @ do fim desta sequência, já preparado com a informação que corresponde ao estado sucessor e direção tomada a partir da antiga configuração atual de $\mathcal{M}$.

Agora, se ia para a esquerda, o procedimento é parecido. Do @ final, o sucessor é um estado que vai levando a cabeça para a esquerda até encontrar um @. Passa para outro estado, ainda levando a cabeça para a esquerda, até encontrar outro @. Depois que o encontra, o restante se dá igual ao que foi descrito no caso do movimento para a direita.

Logo vemos que, apesar de precisar de muito mais estados do que tinha $\mathcal{M}$, a nova máquina possui um número finito de estados. Definimos que seus estados de aceitação e de rejeição, fora na situação de descarte, só aparecem logo após a cabeça imprimir um @. Na nova máquina, o estado de parada devido só aparecerá após a impressão da sequência no último espaço.

Visto tudo isso, fica claro que os símbolos de Γ serão devidamente codificados na nova máquina, até que ela chegue à aceitação ou a rejeição para uma cadeia $\langle w \rangle$. Assim, a saída que obtemos a partir de uma entrada $\langle w \rangle$ é a simples codificação da saída $f(w)$ sobre o alfabeto Σ_{e}. ■

Em posse desse resultado, trazemos uma definição que nos auxiliará.

Definição 19. Sejam f uma função computável e f_{e} uma função, também computável, definida como está no enunciado do teorema anterior. Então, se $\mathcal{M}$ é uma máquina que computa f, chamaremos de $\mathcal{M}_{\mathsf{e}}$ a máquina como está na demonstração do mesmo teorema e, portanto, que computa f_{e}.

Teorema 9.1.2. *Seja f uma função computável e $\mathcal{M}$ uma máquina de Turing que a compute. Existe uma máquina de Turing que, a partir de uma entrada $\langle \mathcal{M} \rangle$, dá como saída $\langle \mathcal{M}_e \rangle$.*

Demonstração. Mostremos esta máquina. Chamemo-la de $\mathcal{E}$. Ela verifica os símbolos existentes em $\mathcal{M}$ e, para cada um, com exceção do $\sqcup$, após o fim da fita, imprime uma sequência de 0 e 1 seguindo a ordem crescente do sistema binário posicional, primeiro sem colocar símbolos 0 à esquerda.

Já mostramos conseguir fazer estas sequências consecutivas, tal como na demonstração do teorema 4.2.1, sobre máquinas não-determinísticas. A diferença é que não estamos trocando uma sequência pela sua sucessora, mas colocando a sucessora ao lado da sua antecessora. Aqui podemos, a cada impressão de uma sequência, imprimir uma cópia dela à sua direita, e sobre esta cópia fazemos a sequência consecutiva. Feita, fazemos uma cópia dela à sua direita, e seguimos, checando se não ultrapassamos o número de símbolos do alfabeto da fita de $\mathcal{M}$ sem o $\sqcup$.

Quando chegarmos ao fim, se a quantidade de dígitos desta última sequência é K, e se já não estamos nela, preenchemos com mais sequências consecutivas até chegar àquela formada por exatamente K símbolos 1. Depois preenchemos as sequências anteriores, indo para trás, com símbolos 0 à esquerda a partir daquela que verificarmos que tem $(K-1)$ símbolos. Checamos quantos símbolos cada uma tem e preenchemos com os 0 a fim de completar K símbolos.

Para que consigamos fazer isto, quando não houver quadrados vazios suficientes para preencher com símbolos 0, iremos mover a porção dali até o fim da fita em algumas unidades para a direita. Assim teremos quadrados livres para podermos completar cada sequência binária a fim de que tenham K dígitos.

Já temos o número de símbolos do alfabeto de entrada registrado na descrição de $\mathcal{M}$. A partir disso, vamos formando as instruções de $\mathcal{M}_{\mathsf{e}}$. As primeiras são as mesmas para quaisquer máquina $\mathcal{M}_{\mathsf{e}}$, da primeira varredura da entrada, apenas tendo que levar em conta a quantidade K de símbolos a cada sequência entre dois @'s consecutivos.

As próximas são da segunda varredura, para checar que na fita de $\mathcal{M}_{\mathsf{e}}$ só há símbolos do alfabeto de entrada de $\mathcal{M}$ sendo representados. Para isto, construímos $\left(2^K+1\right)$ estados. O estado do primeiro quadrado da sequência visa checar se ali há 0 ou 1. Se 0, ele passa para um estado que vai simplesmente levar a cabeça para a direita até encontrar o @ e recomeçar o ciclo se não encontrar, à direita dele, um espaço vazio. Se ali há 1, a máquina passa para outro estado.

Este também visa checar se ali há 0 ou 1. A partir deste momento, o estado sucessor ou é um ou é outro, se ou 0 ou 1. E isto é determinado pela comparação feita pela $\mathcal{E}$ entre as sequências binárias escritas e o tamanho do alfabeto de entrada de $\mathcal{M}$. Os estados agora construídos vão afunilando o reconhecimento para que, no último dos sucessores aponte qual das sequências está ali. Se é uma das sequências posterior ao número de símbolos de tal alfabeto de entrada, o estado levará aos estados de descarte. Se não, ele repetirá o ciclo até que encontremos um espaço vazio.

No espaço vazio, construímos os dois estados que levam de volta ao início da fita. E então construímos os estados que visam fazer o trabalho como na transição da máquina $\mathcal{M}$. Primeiro, os K estados que sucedem o estado inicial no caso da entrada vazia, que imprime $\textvisiblespace$, seguidos do estado que imprime @.

Agora, sejam m o número de elementos do alfabeto da fita de $\mathcal{M}$ e n o seu número de estados. Consultando as sequências binárias registradas no início, para cada estado de $\mathcal{M}$, construímos $\left(1+2+2^2+\cdots+2^K\right)+K=\left(2^{K+1}-1\right)+K=2^{K+1}+K-1$ estados, sendo que $(K+1)$ deles passaram de um @ ao próximo para, neste, fornecer a informação de qual seria a configuração atual de $\mathcal{M}$.

A parte calculada entre parênteses (das potências de base 2) diz respeito ao

escaneamento de 0 ou 1. As K partes calculadas fora dos parênteses dizem respeito aos K estados usados quando o símbolo escaneado à direita do @ era $\llcorner\!\lrcorner$, até que se percorra os K quadrados e se possa imprimir um @ à direita deles.

Em verdade, alguns destes $\left(2^{K+1}+K-1\right)$ estados serão supérfluos em algumas máquinas $\mathcal{M}_{\mathrm{e}}$, pois haverá $\left(2^{K}-m-1\right)$ das sequências binárias registradas que não corresponderão a nenhuma sequência a se fazer presente após verificada uma entrada legítima.

Como sucessor das sequências de estados relativas a estes $\left(2^{K}-m-1\right)$ símbolos inexistentes em $\mathcal{M}$, colocamos o estado de rejeição, embora ele não vá ser realmente posto em prática neste contexto. Para cada um dos m estados que são os últimos de uma sequência das apontadas acima, desta vez correspondentes a um símbolo de $\mathcal{M}$, colocamo-los para fazer a máquina imprimir @ e ir para a esquerda. Cada um deles é sucedido por outro estado que carrega a informação da configuração e que procura o @ mais próximo à esquerda.

Quando o encontra, passa para uma sequência de K estados com que $\mathcal{M}_{\mathrm{e}}$ vai imprimir a sequência binária ou de $\llcorner\!\lrcorner$ do que seria o símbolo a ser impresso em $\mathcal{M}$, os quais definimos pela consulta da sequência binária registrada correspondente. Elas também carregam a informação do que seria o estado sucessor em $\mathcal{M}$ e da direção tomada. Para isso as instruções da entrada recebida por $\mathcal{E}$ são consultadas.

Se a transição do momento exigiria o deslocamento para a direita, os mesmos tipos de estados descritos acima já bastam. Mas ainda falta construir os estados que reproduzem o movimento para a esquerda. Na realidade, o que precisamos são mais dois estados para cada estado de $\mathcal{M}$. Um para encontrar o primeiro @ à esquerda, e um segundo para encontrar o segundo, ambos carregando a informação do estado de $\mathcal{M}$. E a continuidade é igual.

Os estados de aceitação e de rejeição são definidos e sucedem tal como em $\mathcal{M}$, após a mudança de uma sequência de K quadrados e a impressão de @.

Usando este método, consultando as instruções recebidas, as sequências binárias montadas e comparando-as com a quantidade de símbolos que tem o alfabeto de $\mathcal{M}$, conseguimos, com $\mathcal{E}$, construir a descrição da máquina $\mathcal{M}_{\mathrm{e}}$. ■

Observação 15. Lembremos de algo que poderia passar despercebido. Do modo como definimos as máquinas $\mathcal{M}_{\mathrm{e}}$, ela também possui uma versão correspondente a uma máquina do tipo $\mathrm{bg}\left(\mathcal{N}\right)$ para uma máquina de Turing $\mathcal{N}$ qualquer, ou seja, uma máquina $\left(\mathrm{bg}\left(\mathcal{N}\right)\right)_{\mathrm{e}}$. Mas esta última não termina de fato no primeiro quadrado de sua própria fita.

O que não quer dizer que não possamos construir uma máquina equivalente que faça isso. Na verdade é bem simples. Tomando uma máquina $\mathcal{M}_{\mathrm{e}}$ qualquer, como as entradas legítimas sempre começam por @, basta criar os estados que procurem dois símbolos @ seguidos à esquerda, e teremos encontrado o início da fita. Eles

substituem os estados de parada e são sucedidos por eles. A esta máquina modificada com este intuito damos o nome de bge($\mathcal{M}_{\mathsf{e}}$).

Pois bem, anteriormente havíamos falado, a partir de uma função $f : \Sigma_1^* \to \Sigma_2^*$ correlacionada a uma máquina de Turing, em unir os alfabetos presentes no domínio e contradomínio em um alfabeto de entrada Σ para termos uma nova função, esta, computável. Mas dependendo das finalidades, é útil acrescentar mais símbolos para formar Σ do que apenas aqueles da referida união.

Uma situação que requer algo assim seria aquela em que procuramos montar uma máquina que lide com codificações de certos objetos como se fosse uma coordenada dentre outras que a cadeia de entrada pode fornecer. Fosse, neste sentido, Σ' um alfabeto do qual queremos obter entradas quaisquer e Σ'' um alfabeto a partir do qual obtemos objetos P de uma certa classe, sendo ambos alfabetos diferentes, com intersecção não-vazia.

Digamos que tenhamos o objetivo de trabalhar com entradas do tipo $(u, \langle P\rangle)$, com $u \in (\Sigma')^*$. Acontece que, se o domínio da função por trás desta máquina fosse $(\Sigma')^* \times (\Sigma'')^*$, o método que vínhamos adotando para primeiro transformar em função computável já não serve, pois aqui o domínio não é nem sequer um alfabeto sob a operação estrela.

Como ao alfabeto de entrada têm que pertencer todos os símbolos presentes tanto em u como em $\langle P\rangle$, então tal alfabeto conteria ao menos a união de Σ' com Σ''. Só que, sendo assim, se torna mais difícil, para certas entradas, elaborar um algoritmo que distinga quando há um par tal como proposto acima.

E, se é conveniente que haja uma separação entre u e $\langle P\rangle$, se ela ficasse a cargo de um símbolo de Σ', então ficaria difícil distinguir se ele não faz parte das cadeias desejadas como u. Semelhante seria se tomássemos um símbolo de Σ'' para esta tarefa.

Pensando em situações como esta, adotamos a seguinte convenção. Para quando quisermos possuir uma cadeia que é separável em determinadas partes menores, ambas pertencendo a um conjunto $\left(\Sigma^{\#}\right)^*$, vamos trabalhar com algum elemento $\gamma \notin \Sigma^{\#}$ e montar o alfabeto $\Sigma^{\#} \cup \{\gamma\}$.

Trataremos de situações nas quais houver uma cadeia w divisível de forma que $w = w^{(1)}\gamma w^{(2)}$, onde temos $w^{(1)}, w^{(2)} \in \left(\Sigma^{\#}\right)^*$. E, para isto, usaremos a notação de que $w = \left\langle w^{(1)}, w^{(2)}\right\rangle$. Na verdade, este símbolo γ está fazendo exatamente o papel da vírgula. Só não usamos como símbolo a própria vírgula porque ela poderia já estar presente no alfabeto $\Sigma^{\#}$, então tomamos um representante genérico. Mas podemos, em geral, pensar que o símbolo se trata da própria vírgula.

Esta notação pode ser estendida, se quisermos. Teríamos, para $s > 1$ natural, com $i = 1, 2, \ldots, s$ e $w^{(i)} \in \left(\Sigma^{\#}\right)^*$, que $\left\langle w^{(1)}, w^{(2)}, \ldots, w^{(s)}\right\rangle = w^{(1)}\gamma w^{(2)}\gamma \ldots \gamma w^{(s)}$. Enfim, o alfabeto de entrada que teremos para uma máquina com esse objetivo será

$\Sigma = \Sigma^{\#} \cup \{\gamma\}$. E é este que será convertido a partir de Σ_{e}.

Sendo assim, finalizamos esta seção com duas observações.

Observação 16. Resumindo, conseguimos ver que funções de Σ_1^* em Σ_2^* que possuírem uma máquina de Turing que as computa podem ser tratadas a partir de uma função computável, de Σ^* em Σ^*.

Como também vemos que funções de $(\Sigma')^* \times (\Sigma'')^*$ em Σ_2 com uma máquina de Turing que computa suas saídas a partir de entradas que representam seus elementos do domínio, com o auxílio de um símbolo exterior γ, também o podem. Utilizaríamos mecanismos de descarte para entradas que não da forma $w = \langle w^{(1)}, w^{(2)} \rangle$ onde $w^{(1)} \in (\Sigma')^*$ e $w^{(2)} \in (\Sigma'')^*$.

Observação 17. Todas estas funções acima são convertíveis em funções computáveis de Σ_{e}^* em Σ_{e}^*. Sendo assim, iremos considerar, quando trabalharmos com funções como apresentadas acima, que elas recaem em funções computáveis.

9.2 HISTÓRIAS DE COMPUTAÇÃO

Sejam $\mathcal{M}$ uma máquina de Turing e w uma entrada possível dela. Se tal máquina para sobre esta entrada, chamamos de **história de computação** para $\mathcal{M}$ sobre w a sequência de configurações completas (ou seja, cadeia da fita + estado + quadrado escaneado) por que passou esta máquina a partir desta entrada, desde a primeira, em ordem, até última.

Obviamente, para algum $l \in \mathbb{N} - \{0\}$, sendo esta sequência $(C_1, C_2, \ldots, C_l)$, o elemento C_1 se trata da configuração completa inicial. Se $\mathcal{M}$ aceita w, o que temos é a **história de computação de aceitação** para $\mathcal{M}$ sobre w, e assim, C_l é uma configuração completa de aceitação. E se $\mathcal{M}$ rejeita w, o que temos é a **história de computação de rejeição** para $\mathcal{M}$ sobre w, e assim, C_l é uma configuração completa de rejeição

Deste modo, podemos afirmar que máquinas de Turing determinísticas têm, no máximo, uma história de computação para cada entrada. Já as máquinas de Turing não-determinísticas, para cada entrada, podem ter mais de uma, as quais corresponderiam aos ramos finitos dela. Trouxemos tal conceito pois, a partir de histórias de computação, também conseguimos fazer reduções de problemas envolvendo decidibilidade.

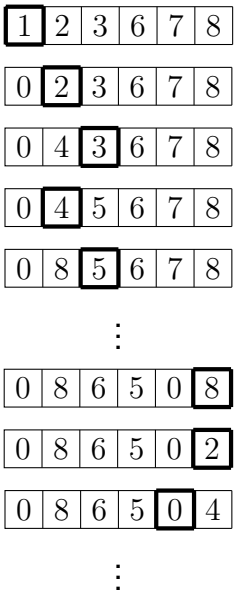
Um primeiro problema desses a apresentarmos trata de um tipo restrito de máquinas, os **autômatos linearmente limitados**, que em quase tudo funcionam como as máquinas de Turing convencionais. Nestas máquinas a diferença é que a sua cabeça não se move para os quadrados de fora da entrada, seja para a esquerda (como

já era comum), seja para a direita (o que é novo). Se a cabeça fosse ultrapassar esses limites, no próximo estágio ela continuaria no mesmo quadrado limítrofe.

Podemos abreviar seu nome para ALL. Assim, se um ALL recebe, por exemplo, uma entrada de quatro símbolos, sua cabeça só vai se mover pelos quatro primeiros quadrados da fita. Se ele recebe uma entrada de dez símbolos, a cabeça só se moverá pelos dez primeiros quadrados.

Exemplo 14. Tomemos, para exemplificar, um ALL cujo alfabeto da fita é o conjunto dos algarismos do sistema decimal e que recebe a entrada 123678.

Figura 102 – Exemplo de computação de um ALL



Fonte: Autor (2024)

Os ALL's podem mover a cabeça tanto para a esquerda quanto para a direita ao longo de seus estados. Perceba ainda que, da antepenúltima para a penúltima fita representadas no exemplo, o que ocorreu foi que a cabeça andaria para a direita, mas chegou ao limite do fim da fita. Permaneceu no mesmo quadrado, portanto, mas com o símbolo que por último fora impresso. E essa máquina continuará trabalhando até chegar ou ao estado de aceitação, ou ao de rejeição. Pode entrar em loop também. □

A memória de um ALL para uma entrada w específica é limitada, no sentido de que, a cada estágio, ele poderá registrar um número finito de configurações completas em sua fita. Em torno dessa afirmação, temos o seguinte resultado.

Lema 9.2.1. *Seja B um ALL que possui q estados e g símbolos no alfabeto da fita. Se B está sobre uma entrada de comprimento k, então existem exatamente qkg^k configurações completas distintas possíveis.*

Demonstração. A máquina, em um estágio qualquer, vai se encontrar em um estado dentre q distintos, com a cabeça em um quadrado dentre k distintos, e com uma cadeia sobre a fita.

Esta cadeia tem comprimento k, sendo que em cada uma de suas k posições pode haver qualquer um dos g símbolos do alfabeto da fita. Assim, há g^k cadeias possíveis na fita.

Sendo esses três itens simultâneos e considerados sem restrições para a configuração completa, a quantidade total dessas últimas é o produto da quantidade de cada um dos três. Portanto, há exatamente qkg^k configurações completas distintas possíveis. ■

Agora, seja Σ um alfabeto capaz de codificar os objetos que vêm a seguir. Seja, então, a linguagem

$$\mathsf{A}_{\mathsf{ALL}} = \{\langle B, w\rangle \in \Sigma^*;\ B \text{ é um ALL que aceita a cadeia de entrada } w\}.$$

Sendo os ALL's muitíssimo semelhantes às máquinas de Turing, um modo de os codificarmos se dá tal como vínhamos mostrando usualmente. Isto é, utilizando os símbolos D, A, C, B, ..., com alguma distinção a fim de que não ficasse igual à codificação das máquinas de Turing.

Conseguimos simulá-los a partir de uma máquina de Turing semelhante à máquina universal. Para garantir que tal máquina fará bem este trabalho, satisfazendo a condição de que a cabeça virtual do ALL simulado não ultrapassará o limite da direita da entrada, podemos fazer o seguinte.

Quando simulávamos máquinas de Turing, nós havíamos deixado o I no início das configurações completas trazidas para o final da fita da máquina $\mathcal{U}$, a fim de não ultrapassar o limite da esquerda da máquina simulada. Similarmente, deixaremos um F ao final das configurações completas na máquina simuladora presente. Quando o quadrado escaneado fosse o posterior ao fim, tal configuração será apagada e o quadrado escaneado será o mesmo de antes, o último da direita.

Proposição 9.2.2. *A linguagem A_{ALL} é decidível.*

Demonstração. Como um ALL funciona de forma quase igual a uma máquina de Turing, parte do que faremos é, dentro de uma máquina de Turing, simulá-lo com a entrada respectiva.

Agora, denotemos por q o número de estados do ALL a ser simulado, por g o número de símbolos do seu alfabeto da fita, e por k o comprimento da entrada w sobre a qual ele está. Assim sendo, outra parte do que faremos consiste em limitar essa simulação a qkg^k estágios de funcionamento do ALL. Isso porque, se o ALL não parar até esse número, como ele está sempre reduzido àqueles quadrados, se ele

repetir uma configuração completa em algum momento, ele entrará em um ciclo de mesmas configurações completas.

E, entrando neste ciclo, dele não sairá. Mas como há exatamente qkg^k configurações completas distintas, se até findar esse número ele não passou por um estado de aceitação ou de rejeição, então ele não passará mais. Ou seja, entrou em loop.

Tendo isso em mente, construamos a máquina de Turing $\mathcal{L}$, como descrita abaixo.

$\mathcal{L}$ = **"** Sobre a entrada $\langle B, w\rangle$, onde B é um ALL e w é uma cadeia:

1. Simule B sobre w por qkg^k estágios dele ou até que ele pare.
2. Se B parou, então aceite se B aceitou e rejeite se B rejeitou. E se B não parou, então rejeite.**"**

Quando B aceita w, consequentemente $\mathcal{L}$ aceita a entrada $\langle B, w\rangle$. Quando B rejeita w, consequentemente $\mathcal{L}$ rejeita a entrada $\langle B, w\rangle$. E quando B entra em loop sobre w, consequentemente $\mathcal{L}$ rejeita $\langle B, w\rangle$.

Sendo assim, $\mathcal{L}$ somente aceita as entradas $\langle B, w\rangle$ quando B aceita w, rejeitando todas as outras entradas. Isto é, $\mathcal{L}$ decide a linguagem $\mathsf{A_{ALL}}$. Portanto $\mathsf{A_{ALL}}$ é decidível. ■

Este resultado nos mostra que a versão do *accepting problem* para ALL's é decidível, ao contrário de como é com as máquinas de Turing. Agora, nem tudo é diferente. O problema de vacuidade dos ALL's, assim como o das máquinas de Turing, também é indecidível. Para o constatarmos, definimos a linguagem

$$\mathsf{V_{ALL}} = \{\langle B\rangle \in \Sigma^*;\ B \text{ é um ALL e } L(B) = \varnothing\}.$$

Feito isso, demonstremos a nossa afirmação. E o faremos reduzindo o problema de vacuidade dos ALL para o *accepting problem* das máquinas de Turing, através de histórias de computação.

Proposição 9.2.3. *A linguagem V_{ALL} é indecidível.*

Demonstração. Mostraremos que, se a linguagem $\mathsf{V_{ALL}}$ fosse decidível, então a linguagem $\mathsf{A_{MT}}$ também o seria, o que contradiz o teorema 5.4.5. Para mostrá-lo, construiremos a máquina hipotética que decide esta última.

Supondo que $\mathsf{V_{ALL}}$ seja decidível, existe uma máquina $\mathcal{R}$ que decide tal linguagem. Isto é, $\mathcal{R}$ aceita somente as entradas $\langle B\rangle$ em que B é um ALL e $L(B) = \varnothing$, e rejeita todas as outras. Usaremos essa máquina a seguir.

Agora, sejam $\mathcal{M}$ uma máquina de Turing qualquer e w uma cadeia de seu alfabeto de entrada. Construiremos, a seguir, um ALL denotado por $B[\mathcal{M}, w]$, do qual a linguagem é o conjunto composto por um único elemento: a história de computação de aceitação da máquina $\mathcal{M}$ sobre w.

Fica claro que $\mathcal{M}$ não aceita w, se, e somente se $L\left(B[\mathcal{M}, w]\right) = \varnothing$. Este ALL verificará se a sua entrada é a sequência de configurações completas que leva $\mathcal{M}$, desde seu estado inicial, no início da cadeia de entrada, até o estado de aceitação.

Para $l \in \mathbb{N} - \{0\}$, denote por $\langle C_1, C_2, \ldots, C_l \rangle$ a codificação de uma história de computação (arbitrária) para $\mathcal{M}$. Teremos que $B[\mathcal{M}, w]$ será de fato um ALL pois ele nunca precisará sair dos limites da entrada que recebe para fazer o seu trabalho, como veremos.

O nosso ALL deve verificar se a C_1 é a configuração completa inicial de $\mathcal{M}$ sobre w; se, para $i \in \mathbb{N}$, com $1 \leq i \leq l-1$, cada C_{i+1} é a configuração completa que segue imediatamente de C_i; e se C_l é uma configuração completa de aceitação para $\mathcal{M}$.

Tomemos a nossa entrada ou como $w = w_1 w_2 \ldots w_k$, ou como um espaço vazio se $w = \varepsilon$, e também $\mathfrak{q}_0$ como o estado inicial de $\mathcal{M}$. Para tal, deixamos na própria constituição de $B[\mathcal{M}, w]$ que ele deve reconhecer se a primeira configuração codificada corresponde ou a $\mathfrak{q}_0 w_1 w_2 \ldots w_k$, ou a $\mathfrak{q}_0 \textvisiblespace$ no caso da cadeia vazia. Como essa configuração é certa, a necessidade da leitura dela é deixada esquematizada ao se construir o ALL.

Depois, a partir da função de transição de $\mathcal{M}$, o ALL deve verificar as configurações completas são de fato sucessoras uma da outra. Para isto, deve verificar se os símbolos nas posições dos quadrados entre C_i e C_{i+1} são as mesmas, com exceção da posição sobre a qual a cabeça de $\mathcal{M}$ estava em C_i, e também das posições adjacentes a esta.

Nestas últimas posições, a verificação deverá ser a respeito de se a função de transição de $\mathcal{M}$ foi devidamente aplicada. Deste modo, $B[\mathcal{M}, w]$ deve ter, em sua constituição, para quando chegar este momento, a informação das transições que $\mathcal{M}$ faria de acordo com o estado e com o símbolo escaneado.

Com isso, neste passo, $B[\mathcal{M}, w]$ está diante de determinado estado, e em determinado símbolo escaneado em C_i, ou seja, está diante de uma configuração. Aquela mesma posição de quadrado já teria, em C_{i+1}, o símbolo que consta na imagem desta configuração através da função de transição. Além de, em C_{i+1}, o novo quadrado escaneado ser aquele para o qual a direção correta levava a cabeça de $\mathcal{M}$. Como também, o estado sucessor correto deve ali estar.

O ALL deve então, nas posições em geral, comparar, um por um, os símbolos de tais posições, entre uma configuração completa e outra. Para registrar as posições já verificadas, podemos usar o recurso de trocar o símbolo que ali estava por sua

versão com um ponto em cima. Assim, se estivermos adotando a codificação por meio dos símbolos D, A, C, ..., neste passo iríamos substituir alguns deles por Ḋ, Ȧ, Ċ, etc.

E, após tudo isto, se tal história de computação estiver de acordo, resta a $B[\mathcal{M}, w]$ verificar se a C_l é realmente uma configuração completa de aceitação. Como a condição de as configurações completas serem as devidas sucessoras uma da outra foi satisfeita, tudo o que é necessário fazer é ver se o estado presente em C_l é o estado de aceitação. Se o for, então tal história de computação está na linguagem de $B[\mathcal{M}, w]$. Se não o for, então não está.

É possível ver que conseguimos construir o ALL $B[\mathcal{M}, w]$ a partir de uma máquina de Turing que receba a entrada $\langle \mathcal{M}, w \rangle$, seguindo o roteiro dos passos dados acima. E, tendo isso em mente, construiríamos a máquina de Turing $\mathcal{S}$ como descrita abaixo.

$\mathcal{S} =$ " Sobre a entrada $\langle \mathcal{M}, w \rangle$, onde $\mathcal{M}$ é uma máquina de Turing e w é uma cadeia:

1. A partir de $\mathcal{M}$ e w, construa o ALL $B[\mathcal{M}, w]$ conforme descrito anteriormente.
2. Emule a máquina $\mathcal{R}$ sobre a entrada $\langle B[\mathcal{M}, w] \rangle$.
3. Se $\mathcal{R}$ rejeita, então aceite. Se $\mathcal{R}$ aceita, então rejeite."

Se $\mathcal{R}$ rejeita $\langle B[\mathcal{M}, w] \rangle$, isso significa que $L\left(B[\mathcal{M}, w]\right) \neq \varnothing$. Consequentemente, $B[\mathcal{M}, w]$ tem uma história de computação de aceitação, o que implica que $\mathcal{M}$ aceita w. E é nesta situação que a máquina $\mathcal{S}$ está aceitando a entrada $\langle \mathcal{M}, w \rangle$.

Se $\mathcal{R}$ aceita $\langle B[\mathcal{M}, w] \rangle$, isso significa que $L\left(B[\mathcal{M}, w]\right) = \varnothing$. Consequentemente, $B[\mathcal{M}, w]$ não tem uma história de computação de aceitação, o que implica que $\mathcal{M}$ não aceita w. E nesta situação a máquina $\mathcal{S}$ rejeita $\langle \mathcal{M}, w \rangle$.

Sendo assim, $\mathcal{S}$ aceita $\langle \mathcal{M}, w \rangle$ se, e somente se $\mathcal{M}$ aceita w. Isto é, $\mathcal{S}$ decide A_{MT}. Como isto não é verdadeiro, segue que $\mathsf{V}_{\mathsf{ALL}}$ é indecidível. ■

Este recurso das histórias de computação será utilizado novamente no próximo capítulo, dedicada a um problema específico.

10 O PROBLEMA DA CORRESPONDÊNCIA DE POST

Eis que, então, trazemos um resultado importante, a fim de nos ajudar a expandir nossa visão quanto ao alcance das máquinas de Turing para resolver certas situações. Dá-se em torno de um problema que, levando o nome do matemático Emil Post, por si só, não está ligado a máquinas de Turing ou semelhantes, mas que também é indecidível.

10.1 ESTABELECENDO O PROBLEMA

Descrevemos, a seguir, o chamado **problema da correspondência de Post**, ou simplesmente **PCP**. Tomamos sempre um conjunto de dominós, nos quais suas duas pontas possuem, cada, uma cadeia, como abaixo.

Figura 103 – Exemplar de dominó

$$\left[\frac{a}{ab}\right]$$

Fonte: Autor (2024)

Olhemos para um conjunto específico de dominós, dado a seguir.

$$\left\{\left[\frac{d}{db}\right], \left[\frac{bd}{dd}\right], \left[\frac{dbca}{ca}\right]\right\}$$

Eis que jogaremos um jogo. Seu objetivo, usando esse conjunto, é procurar uma sequência finita de seus dominós, podendo repeti-los, de modo que a cadeia lida na parte de cima dessa sequência seja igual à cadeia lida na parte de baixo dela. Chamamos esta sequência de **emparelhamento**. E, com o conjunto de dominós acima, conseguimos fazer o emparelhamento abaixo.

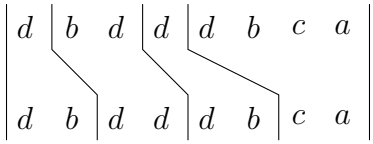
Figura 104 – Dominós do conjunto emparelhados

$$\left[\frac{d}{db}\right]\left[\frac{bd}{dd}\right]\left[\frac{d}{db}\right]\left[\frac{dbca}{ca}\right]$$

Fonte: Autor (2024)

A cadeia de cima, ao final, é $dbdddbca$. E a cadeia de baixo também. Com os dominós intactos, os símbolos de cima não ficam exatamente alinhados ao seu correspondente de baixo. Podemos redesenhar tal emparelhamento para que fiquem alinhados.

Figura 105 – Emparelhamento desenhado



Fonte: Autor (2024)

Esta forma de representar será usada na demonstração da indecidibilidade posteriormente. Mas conseguimos representar este emparelhamento porque, para o conjunto em questão, era possível. Há conjuntos de dominós para os quais não há emparelhamento. A seguir um exemplo.

$$\left\{ \left[\frac{d}{db}\right], \left[\frac{a}{ad}\right], \left[\frac{ab}{dba}\right] \left[\frac{dd}{bba}\right] \right\}$$

Para este conjunto de dominós não há emparelhamento possível porque as cadeias de baixo são sempre mais longas que as cadeias de cima. Assim, qualquer sequência teria a parte de baixo maior que a de cima, e as duas nunca seriam iguais.

Havíamos dito que o objetivo do jogo era procurar um emparelhamento, e não necessariamente encontrá-lo, justamente porque por vezes ele não existe. Podemos dizer que a pessoa vence o jogo com um conjunto de dominós se conseguir concluir a respeito de tal conjunto ter um emparelhamento.

É justamente nisso que consiste o problema da correspondência de Post: descobrir quando uma coleção de dominós possui emparelhamento e quando não possui.

Como já adiantamos, tal problema é indecidível. Ele não possui uma solução com algoritmos. Mostrá-lo-emos através da versão dele para linguagens. Para a definir, vejamos que as entradas a serem consideradas por ele, isto é, suas **instâncias**, serão conjuntos P de dominós,

$$P = \left\{ \left[\frac{t_1}{s_1}\right], \left[\frac{t_2}{s_2}\right], \ldots, \left[\frac{t_p}{s_p}\right] \right\},$$

onde $p \in \mathbb{N} - \{0\}$, e, para $i \in \mathbb{N}$, $1 \leq i \leq p$, temos que t_i e s_i são sequências não-vazias.

Um emparelhamento fica definido como uma sequência finita $(i_1, i_2, \ldots, i_l)$ de números naturais de 1 a p, tais que $t_{i_1} t_{i_2} \ldots t_{i_l} = s_{i_1} s_{i_2} \ldots s_{i_l}$. E o problema da correspondência de Post acaba trabalhando com a determinação de se P possui ou não um emparelhamento.

Para isso, definimos então uma linguagem. Através de uma determinada maneira de codificar, teremos

$$\text{PCP} = \{\langle P \rangle \in \Sigma_{\mathsf{e}}^*;\ P \text{ é uma instância do problema da}$$

correspondência de Post que tem um emparelhamento$\}$.

O que faremos a seguir é demonstrar, através de uma redução, que o problema da correspondência de Post é indecidível. Provaremos que o *accepting problem* é redutível por mapeamento ao problema da correspondência de Post. No entanto, para concluirmos isto, utilizaremos uma linguagem intermediária, que fará a transição entre a $\mathsf{A_{MT}}$ e a PCP.

Então, antes de irmos para a próxima seção, onde demonstraremos o teorema que aborda essa questão, daremos alguns passos importantes.

Lema 10.1.1. *Seja $\mathcal{M}$ uma máquina de Turing qualquer. Existe uma máquina de Turing $\mathcal{K}$ que, ao ler uma entrada $\langle\mathcal{M}\rangle$, fornece $\langle R(\mathcal{M})\rangle$ como saída, onde $R(\mathcal{M})$ é uma máquina de Turing correspondente a $\mathcal{M}$, de modo que $R(\mathcal{M})$ nunca tenta andar para a esquerda do início de sua fita.*

Demonstração. Seja $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, \mathfrak{q}_0, \mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}})$ uma máquina de Turing qualquer, com $Q = \{\mathfrak{q}_0, \mathfrak{q}_1, \ldots, \mathfrak{q}_{n-1}\}$ e $\Gamma = \{\alpha_0, \alpha_1, \ldots, \alpha_{m-1}\}$. Está fixado que $\alpha_0 = \text{␣}$.

Se o estado inicial de $\mathcal{M}$ é de aceitação ou de rejeição, podemos considerar $\mathsf{R}(\mathcal{M}) = \mathcal{M}$, pois $\mathcal{M}$ já é tal que não vai andar para a esquerda a partir do primeiro quadrado. Isto porque ela não fará deslocamento algum.

Já se o estado inicial não é um estado de parada, o que iremos fazer é construir uma máquina de Turing $\mathsf{R}(\mathcal{M})$ que substitui sempre o símbolo do primeiro quadrado por um símbolo correspondente. Esta nova máquina, ao ler tal símbolo, nunca tenta ir para a esquerda. Em vez disto, ela anda para a direita e volta para a esquerda logo em seguida. O funcionamento do restante de suas atividades é mantido, para ter o mesmo efeito que $\mathcal{M}$ teria.

Enfim, para $\mathsf{R}(\mathcal{M})$, tomaremos símbolos, distintos entre si, apresentados no conjunto $\Gamma_1 = \{\alpha_m, \alpha_{m-1}, \ldots, \alpha_{2m-1}\}$. Tomamo-los de modo tal que $\Gamma \cap \Gamma_1 = \varnothing$ e, para $i \in \mathbb{N}$, com $0 \leq i < m$, o símbolo α_{m+i} será o correspondente de α_i.

Também tomaremos uma quantidade de $(n+1)$ novos estados. Sejam $r_1, r_2 \in \mathbb{N}$ tais que $\mathfrak{q}_{r_1} = \mathfrak{q}_{\text{aceita}}$ e $\mathfrak{q}_{r_2} = \mathfrak{q}_{\text{rejeita}}$. Agora tomemos o conjunto $Q' = \{\mathfrak{q}'_0, \mathfrak{q}'_1, \mathfrak{q}'_2, \ldots, \mathfrak{q}'_{2n}\}$. Para $j \in \mathbb{N}$, com $0 \leq j < n$, temos que $\mathfrak{q}'_{j+1} = \mathfrak{q}_j$.

Seja então $\Gamma' = \Gamma \cup \Gamma_1$. Tomaremos a função $\delta' : Q' \times \Gamma' \to Q' \times \Gamma' \times \{\mathsf{R}, \mathsf{L}\}$, cuja lei descrevemos a seguir.

Para qualquer valor de i, para algum $i^{\#} \in \mathbb{N}$, com $0 \leq i^{\#} < m$, e para algum $j^{\#} \in \mathbb{N}$, com $0 \leq j^{\#} < n$, se $\delta(\mathfrak{q}_0, \alpha_i) = \left(\mathfrak{q}'_{j^{\#}}, \alpha_{i^{\#}}, \mathsf{R}\right)$, então teremos $\delta'(\mathfrak{q}'_0, \alpha_i) = \left(\mathfrak{q}_{j^{\#}}, \alpha_{m+i^{\#}}, \mathsf{R}\right)$, e se $\delta(\mathfrak{q}_0, \alpha_i) = \left(\mathfrak{q}_{j^{\#}}, \alpha_{i^{\#}}, \mathsf{L}\right)$, teremos $\delta'(\mathfrak{q}'_0, \alpha_i) = \left(\mathfrak{q}'_{n+j^{\#}+1}, \alpha_{m+i^{\#}}, \mathsf{R}\right)$.

Para quaisquer valores i e j, e com eles valores $i^* \in \mathbb{N}$, com $0 \leq i^* < m$, e $j^* \in \mathbb{N}$, onde $0 \leq j^* < n$, teremos que, se $\delta(\mathfrak{q}_j, \alpha_i) = (\mathfrak{q}_{j^*}, \alpha_{i^*}, \mathsf{R})$, então $\delta'\left(\mathfrak{q}'_{j+1}, \alpha_{m+i}\right) = \left(\mathfrak{q}'_{j^*}, \alpha_{m+i^*}, \mathsf{R}\right)$, e se $\delta(\mathfrak{q}_j, \alpha_i) = (\mathfrak{q}_{j^*}, \alpha_{i^*}, \mathsf{L})$, então $\delta'\left(\mathfrak{q}'_{j+1}, \alpha_{m+i}\right) = \left(\mathfrak{q}'_{n+j^*+1}, \alpha_{m+i^*}, \mathsf{R}\right)$.

Para quaisquer valores i e j, teremos $\delta'\left(\mathfrak{q}'_{n+j+1}, \alpha_i\right) = \left(\mathfrak{q}'_{j+1}, \alpha_i, \mathsf{L}\right)$.

Nos estados $\mathfrak{q}'_{j+1}$, com os símbolos α_i, deixamos que as transições por δ' sejam iguais às transições de $\mathfrak{q}_j$ com os símbolos α_i. Quanto às demais transições destes estados, diremos que serão para o estado $\mathfrak{q}_{\text{aceita}}$, imprimindo ␣, e indo para a direita. Mas, em verdade, elas poderiam ser quaisquer, pois não farão diferença.

Pois bem. A máquina $\mathsf{R}(\mathcal{M}) = \left(Q', \Sigma, \Gamma', \delta', \mathfrak{q}'_0, \mathfrak{q}'_{r_1+1}, \mathfrak{q}'_{r_2+1}\right)$ satisfaz o que procurávamos.

Nos estados $\mathfrak{q}'_0$ e $\mathfrak{q}'_{j+1}$ a máquina, quando no primeiro quadrado, imprime símbolos correspondentes aos α_i que originalmente apareceriam ali, nunca tenta ir para a esquerda, e prossegue, deixando rastro igual ao que deixava $\mathcal{M}$. Quando com tais estados, nos demais quadrados, ela faz a mesma coisa que $\mathcal{M}$ fazia.

E os estados $\mathfrak{q}'_{n+j+1}$ só aparecerão no segundo quadrado, servindo para a cabeça retornar ao primeiro quadrado quando originalmente $\mathcal{M}$ teria tentado ir para a esquerda do início da fita no passo anterior. Então eles sempre lerão apenas símbolos de Γ.

Assim, $\mathsf{R}(\mathcal{M})$ reconhecerá a mesma linguagem que $\mathcal{M}$. Agora, vejamos quanto a $\mathcal{K}$. O dever dela é ajustar a descrição de $\mathcal{M}$ para que se torne $\mathsf{R}(\mathcal{M})$.

No caso em que ela verifica que o estado inicial é um estado de parada, ela simplesmente mantém a descrição de $\mathcal{M}$. Vejamos, então, no outro caso.

Para isso, ela pode verificar o número de símbolos e o número de estados de $\mathcal{M}$, registrando ambos à direita na fita com dois símbolos distintos. Podem ser, digamos, símbolos $\mathbf{u}$ para o número de símbolos e símbolos $\mathbf{v}$ para o número de estados, acrescido de um $\mathbf{x}$ à direita, todos espaçados por um quadrado cada. Consultará esse número representado pela quantidade de $\mathbf{u}$'s e de $\mathbf{v}$'s seguidamente.

Ao fim da fita, $\mathcal{K}$ vai escrevendo as instruções de $\mathfrak{q}'_0$, o estado inicial de $\mathsf{R}(\mathcal{M})$. Elas contam com todos os símbolos escaneáveis, de α_0 a α_{2m-1}, assim como todas as instruções daqui para frente. Para isso, ela primeiro traz o mesmo número de instruções original de $\mathfrak{q}_0$, e depois escreve as outras a partir da comparação/marcação dos símbolos $\mathbf{u}$.

Para fazer as mudanças de acrescentar m símbolos C, e $(n+1)$ símbolos A, a máquina vai utilizando das comparações com $\mathbf{u}$ e com $\mathbf{v}$ e $\mathbf{x}$. Para escrever as instruções dos $(2n+1)$ estados a máquina começa escrevendo as instruções próprias de $\mathfrak{q}'_0$. Depois escreve as instruções dos n seguintes baseadas nas instruções dos n estados de $\mathcal{M}$. E os últimos n novos estados se baseiam na mesma quantidade.

Depois disso, copia os símbolos B para o fim da fita, devidamente espaçados. E então, são apagadas as instruções de $\mathcal{M}$, os $\mathbf{u}$, $\mathbf{v}$, $\mathbf{x}$, completamos os espaços em branco entre as instruções de $\mathsf{R}(\mathcal{M})$ com $\widehat{\text{␣}}$ no lugar. E levamos tudo para o início da fita.

Com isso, vemos que a máquina $\mathcal{K}$ consegue cumprir a tarefa necessária, e o resultado está provado. ■

Através de uma determinada maneira de codificar, sejam os conjuntos

$$MM = \{\langle \mathcal{M}, w\rangle \in \Sigma_{\mathsf{e}}^*;\ \mathcal{M} \text{ é uma máquina}$$

$$\text{de Turing e } w \text{ é uma entrada de } \mathcal{M}\}$$

e

$$PP = \{\langle P\rangle \in \Sigma_{\mathsf{e}}^*;\ P \text{ é uma instância}$$

$$\text{do problema da correspondência de Post}\}.$$

Podemos entender que uma mesma codificação de máquina de Turing diz respeito a inúmeras máquinas de Turing. Todas elas possuindo alfabetos da fita de mesmo tamanho com símbolos de α_0 até α_{m-1}, alfabetos de entrada de mesmo tamanho, mesma quantidade de estados desde $\mathfrak{q}_0$ até $\mathfrak{q}_{n-1}$, mesma posição dos estados inicial, de aceitação e de rejeição e funções de transição tais que levam um par $(\mathfrak{q}_i, \alpha_j)$ até o trio $(\mathfrak{q}_{i'}, \alpha_{j'}, \mathsf{X})$, com $\mathsf{X} \in \{\mathsf{R}, \mathsf{L}\}$.

Ou seja, terão mesma codificação máquinas de Turing que se comportam da mesma maneira, mesmo que com símbolos diferentes. Assim como terão mesma codificação as instâncias do problema da correspondência de Post com dominós semelhantes onde mudam apenas o formato do símbolo em cada posição, mas não a mesma ocorrência de dado símbolo em cada posição.

Definiremos uma linguagem auxiliar para conseguirmos chegar à questão que envolve a decidibilidade do problema da correspondência de Post. Esta linguagem lida, no fim das contas, com o que chamaremos de **problema da correspondência de Post modificado**. Tal problema trata de se uma instância do problema original possui emparelhamento a partir de um de seus dominós específicos.

Assim, se uma dada instância do problema original possui p dominós, a partir dela temos p instâncias distintas do problema modificado a considerar. Para cada uma delas relativa a um dos dominós, de modo a verificarmos se, começando por ele, há emparelhamento. Por isso, fica definida a linguagem

$$\mathsf{PCPM} = \left\{ \left\langle P, \left[\frac{t}{s}\right] \right\rangle \in \Sigma_{\mathsf{e}}^*;\ P \in PP, \left[\frac{t}{s}\right] \in P \text{ e } P \text{ possui}\right.$$

$$\left.\text{um emparelhamento cujo primeiro dominó é } \left[\frac{t}{s}\right] \right\}.$$

10.2 TRABALHANDO EM CIMA DO PROBLEMA

Nossa estratégia será mostrar que um problema indecidível pode ser redutível por mapeamento ao PCP. Este problema será o *accepting problem* das máquinas de Turing. E, para mostrarmos isto, usaremos a linguagem PCPM como intermediária.

Esta seção, então, é totalmente dedicada à demonstração de dois lemas, que implicam no grande teorema da indecidibilidade do PCP.

Lema 10.2.1. *Sendo*

$$A_{MT} = \{\langle \mathcal{M}, w\rangle \in \Sigma_e^*;\ \mathcal{M} \text{ é uma máquina de Turing e } \mathcal{M} \text{ aceita } w\},$$

então $A_{MT} \leq_m$ *PCPM.*

Demonstração. A primeira coisa que faremos é **A)** construir uma função específica de Σ_{e}^* em Σ_{e}^* que leva os elementos de MM em PP. A seguir, mostraremos que **B)**, uma máquina de Turing $\mathcal{M}$ aceita w se, e somente se um elemento $P' \in$ PCPM é imagem de $\langle \mathcal{M}, w\rangle$ através dessa função. Daí, mostraremos que, **C)** a partir dela, formamos uma função computável com a mesma característica. Com isto, teremos provado o lema 10.2.1.

Parte A. Tomando então uma entrada $\langle \mathcal{M}, w\rangle$, a partir dela iremos construir uma instância do problema da correspondência de Post ao final do processo, de modo que as cadeias dos dominós dessa instância visem representar uma história de computação de aceitação de $\mathcal{M}$ sobre w.

Quando o natural $k > 0$ é o comprimento da entrada, seja $w = w_1 w_2 \ldots w_k$. Usando a notação comum para o estado inicial, para conseguirmos fazer o emparelhamento da história de computação que queremos, a configuração completa inicial deve ser $\mathfrak{q}_0 w_1 w_2 \ldots w_k$. Caso $w = \varepsilon$, diremos que a configuração completa inicial é $\mathfrak{q}_0 \sqcup$. De qualquer maneira, isso significa que o início da cadeia emparelhada deve ser fixo.

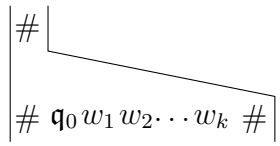
Cabe, ainda, falarmos que tudo o que faremos será usando uma máquina de Turing. Então devemos entender que, quando estivermos falando de um dominó, o que usaremos, na prática, será a codificação dele dentro da máquina. Dito isso, o dominó para começarmos representando a configuração completa inicial de nossa história de computação de aceitação, para o caso de $k > 0$, está na figura 106.

E no caso de $w = \varepsilon$ temos como na figura 107.

Já na máquina, o que realmente teremos são, respectivamente,

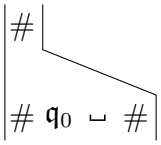
$$\text{ou } \left\langle \left[\frac{\#}{\#\mathfrak{q}_0 w_1 w_2 \ldots w_k \#} \right] \right\rangle, \text{ ou } \left\langle \left[\frac{\#}{\#\mathfrak{q}_0 \sqcup \#} \right] \right\rangle.$$

Figura 106 – Desenho do primeiro dominó para entrada não-vazia



Fonte: Sipser (2005)

Figura 107 – Desenho do primeiro dominó para a entrada vazia



Fonte: Sipser (2005)

Chamemos o dominó apresentado acima de $\left[\frac{t_1}{s_1}\right]$. Requisitamos que os emparelhamentos a serem dispostos comecem por este dominó e não por qualquer um. Por isso, em vez de utilizarmos pura e simplesmente o problema da correspondência de Post, adentraremos agora no problema da correspondência de Post modificado, pois visamos que nossos emparelhamentos iniciem pelo referido $\left[\frac{t_1}{s_1}\right]$.

Dito isso, vamos impor mais um requisito, o qual não comprometerá em nada realmente. Em vez de tomarmos a máquina $\mathcal{M}$ e a partir dela formarmos os dominós, vamos tomar a máquina R$(\mathcal{M})$ indicada no lema 10.1.1.

Sejam $Q = \{\mathfrak{q}_0, \mathfrak{q}_1, \ldots, \mathfrak{q}_{n-1}\}$ o conjunto de estados de R$(\mathcal{M})$, $\mathfrak{q}_{\text{aceita}}$ seu estado de aceitação e $\mathfrak{q}_{\text{rejeita}}$ seu estado de rejeição. O nosso primeiro dominó será

$$\left[\frac{t_1}{s_1}\right] = \begin{cases} \left[\dfrac{\#}{\#\mathfrak{q}_0 w_1 w_2 \ldots w_k \#}\right], & \text{se } k > 0, \\ \left[\dfrac{\#}{\#\mathfrak{q}_0 \sqcup \#}\right], & \text{se } k = 0. \end{cases}$$

Depois disso, devemos ir consultando como se dá o trabalho de R$(\mathcal{M})$ com a entrada w, para daí ir formando e encaixando os próximos dominós.

No momento, já sabemos que, na cadeia de cima do primeiro dominó, devemos prosseguir com a configuração completa inicial. E na cadeia de baixo já podemos passar para a próxima configuração completa. Sabendo disso, definimos como trabalhar com os dominós, e no caso, a cabeça da máquina R$(\mathcal{M})$ irá agora para a direita.

Mas essa mudança só afeta o quadrado escaneado e seus quadrados adjacentes. Então também definimos como trabalhar com os dominós que dizem respeito ao

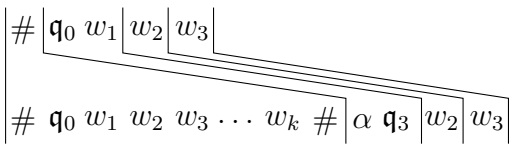
restante dos símbolos da configuração completa.

Sejam Γ e δ respectivamente o alfabeto da fita e a função de transição de R($\mathcal{M}$). Consideremos $\mathfrak{q} \in Q - \{\mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}}\}$. Formamos então, para todos $\beta_1, \beta_2, \beta_3 \in \Gamma$ e para todo $\mathfrak{q}' \in Q$, se $\delta(\mathfrak{q}, \beta_1) = (\mathfrak{q}', \beta_2, \mathsf{R})$, os dominós $\left[\frac{\mathfrak{q}\beta_1}{\beta_2\mathfrak{q}'}\right]$; e se $\delta(\mathfrak{q}, \beta_1) = (\mathfrak{q}', \beta_2, \mathsf{L})$, os dominós $\left[\frac{\beta_3\mathfrak{q}\beta_1}{\mathfrak{q}'\beta_3\beta_2}\right]$. Todos esses dominós serão contados em P'.

E, para preenchermos o restante de símbolos das configurações completas, formamos ainda os dominós $\left[\frac{\beta_1}{\beta_1}\right]$, também contados em P'.

Observemos que todos estes dominós estão em quantidade finita. Para entendermos como seria a continuidade do pretenso emparelhamento, vejamos um pouco de como colocaríamos os dominós para o próximo estágio da máquina R($\mathcal{M}$). Suponhamos, para isso, que, tomando o exemplo da cadeia de entrada não-vazia, tivéssemos $\delta(\mathfrak{q}_0, w_1) = (\mathfrak{q}_3, \alpha, \mathsf{R})$, onde $\alpha \in \Gamma$.

Figura 108 – Desenho do início de um possível emparelhamento com nossos dominós



Fonte: Autor (2024)

Ou seja, sequenciamos, respectivamente, os dominós da figura abaixo.

Figura 109 – Dominós recém-sequenciados (parte 1)

$$\left[\frac{\#}{\#\mathfrak{q}_0 w_1 w_2 w_3 \ldots w_k \#}\right] \left[\frac{\mathfrak{q}_0 w_1}{\alpha \mathfrak{q}_3}\right] \left[\frac{w_2}{w_2}\right] \left[\frac{w_3}{w_3}\right]$$

Fonte: Autor (2024)

Prosseguindo com o nosso empreendimento, também formamos os dominós $\left[\frac{\#}{\#}\right]$ e $\left[\frac{\#}{\sqcup\#}\right]$, contados dentro de P'. Ambos servem para serem colocados ao final de uma configuração completa, de modo a separar a atual da posterior.

O segundo desses dominós é colocado quando o quadrado escaneado pela cabeça de R($\mathcal{M}$) já chegou ao primeiro quadrado vazio após o fim de sua fita. Assim, não deixamos o estado atual sem um símbolo à sua direita na configuração completa. E se deve seguir adicionando tal dominó sempre que tal cabeça persistir em andar para a direita após o fim da fita.

Já o primeiro deles é colocado quando o estado da configuração completa está em qualquer lugar, exceto para além do fim da fita. Assim os dois cumprem seu papel.

Também formamos todos os dominós $\left[\frac{\beta_1 \mathfrak{q}_{\text{aceita}}}{\mathfrak{q}_{\text{aceita}}}\right]$ e $\left[\frac{\mathfrak{q}_{\text{aceita}}\beta_1}{\mathfrak{q}_{\text{aceita}}}\right]$. E por fim, o dominó $\left[\frac{\mathfrak{q}_{\text{aceita}}\#\#}{\#}\right]$.

Não faremos um dominó semelhante a estes a partir do estado de rejeição, e isto é proposital. Os dominós apresentados até aqui são todos os dominós de P', formados a partir de $\mathcal{M}$ e w.

Denominamos então $g : MM \to \Sigma_{\mathsf{e}}^*$ a função para a qual $g(\langle \mathcal{M}, w \rangle) = \langle P' \rangle$, sendo P' o par ordenado constituído da instância do problema da correspondência de Post montada como acabamos de descrever, seguida do dominó $\left[\frac{t_1}{s_1}\right]$.

Parte B. Primeiro vamos provar que **1)** se $\mathcal{M}$ aceita w, então $\langle P' \rangle \in$ PCPM; e em seguida, **2)** a recíproca.

Parte B. 1. Como vínhamos mostrando, após o início do pretenso emparelhamento em $\left[\frac{t_1}{s_1}\right]$, os dominós $\left[\frac{\beta_1}{\beta_1}\right]$, $\left[\frac{\#}{\#}\right]$, $\left[\frac{\#}{\sqcup\#}\right]$ e os dominós das transições em δ iam sequenciando bem. Seguimos completando com os dominós, consultando qual seria a próxima configuração completa de $\mathsf{R}(\mathcal{M})$ a partir de w.

Usamos os dominós com os estados para os movimentos de esquerda e de direita. Usamos os dominós apenas com símbolos para o restante da configuração completa. E os dominós com $\#$ para separar as configurações completas posteriores. Isso até encontrar um estado de parada.

Se em algum momento passarmos para o estado de rejeição, paramos com os dominós e a tentativa de emparelhamento. Mais precisamente, fica claro que não haverá emparelhamento a partir da entrada w em questão. Afinal, não há dominós com o estado de rejeição na cadeia de cima.

E isto vale inclusive para quando o estado inicial é o estado de rejeição. Não haverá como dar continuidade após o primeiro dominó, visto que, sendo $\mathfrak{q}_0 = \mathfrak{q}_{\text{rejeita}}$, não fizemos dominós com $\mathfrak{q}_0$ na cadeia de cima.

Agora, quando se trata do estado de aceitação, teremos formado todos os dominós $\left[\frac{\beta_1 \mathfrak{q}_{\text{aceita}}}{\mathfrak{q}_{\text{aceita}}}\right]$ e $\left[\frac{\mathfrak{q}_{\text{aceita}}\beta_1}{\mathfrak{q}_{\text{aceita}}}\right]$, contados em P'.

A partir do momento em que já temos o estado de aceitação na configuração completa, o que nos falta é terminar de alinhar a cadeia de cima com a cadeia de baixo, pois a história de computação por nós procurada acabou de ser encontrada.

O que os últimos dominós farão é adicionar símbolos que já não representam novas configurações completas para que, após a configuração completa de aceitação debaixo, eles vão diminuindo a diferença de tamanho entre a cadeia de cima e a debaixo.

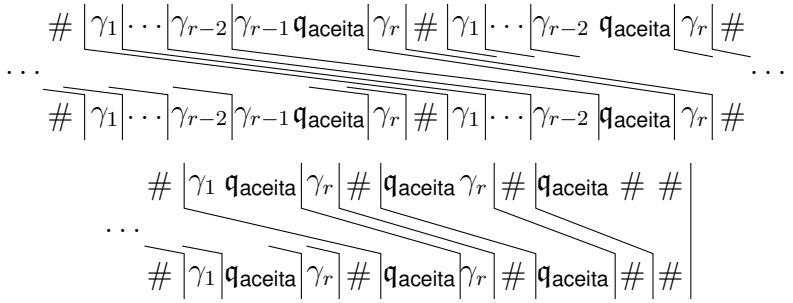
E eles o fazem de modo a prosseguir até logo antes de essa diferença se tornar

nula. Podemos dizer que, com estes dominós, estamos acrescentando pseudopassos à nossa história de computação. Eles, de fato, não estão significando nada em termos da computação da máquina original. Servem apenas com o propósito de concluir o emparelhamento.

Vamos completando o emparelhamento, junto desses dominós, com os dominós $\left[\frac{\beta_1}{\beta_1}\right]$ e $\left[\frac{\#}{\#}\right]$. Para que a diferença de tamanho entre as duas cadeias finalmente se torne realmente nula e fechemos o emparelhamento, adicionaremos o dominó $\left[\frac{\mathfrak{q}_{\text{aceita}}\#\#}{\#}\right]$ ao final.

Para percebê-lo, vejamos com o exemplo abaixo, onde, para um natural $r > 0$, temos $\beta, \gamma_1, \gamma_2, \gamma_{r-2}, \gamma_{r-1}, \gamma_r \in \Gamma$, temos $\overline{\mathfrak{q}} \in Q - \{\mathfrak{q}_{\text{aceita}}, \mathfrak{q}_{\text{rejeita}}\}$ e onde a última transição de $\mathsf{R}(\mathcal{M})$ foi $\delta'(\overline{\mathfrak{q}}, \beta) = (\mathfrak{q}_{\text{aceita}}, \gamma_{r-1}, \mathsf{R})$.

Figura 110 – Desenho da conclusão de um emparelhamento com nossos dominós



Fonte: Autor (2024)

Os dominós que finalizaram este emparelhamento eram, respectivamente, os colocados a seguir.

Figura 111 – Dominós recém-sequenciados (parte 2)

$$\left[\frac{\#}{\#}\right], \left[\frac{\gamma_1}{\gamma_1}\right], \left[\frac{\gamma_2}{\gamma_2}\right], \ldots, \left[\frac{\gamma_{r-2}}{\gamma_{r-2}}\right], \left[\frac{\overline{\mathfrak{q}}\beta}{\gamma_{r-1}\mathfrak{q}_{\text{aceita}}}\right], \left[\frac{\gamma_r}{\gamma_r}\right], \left[\frac{\#}{\#}\right],$$
$$\left[\frac{\gamma_1}{\gamma_1}\right], \left[\frac{\gamma_2}{\gamma_2}\right], \ldots, \left[\frac{\gamma_{r-2}}{\gamma_{r-2}}\right], \left[\frac{\gamma_{r-1}\mathfrak{q}_{\text{aceita}}}{\mathfrak{q}_{\text{aceita}}}\right], \left[\frac{\gamma_r}{\gamma_r}\right], \left[\frac{\#}{\#}\right],$$
$$\left[\frac{\gamma_1}{\gamma_1}\right], \left[\frac{\gamma_2}{\gamma_2}\right], \ldots, \left[\frac{\gamma_{r-2}\mathfrak{q}_{\text{aceita}}}{\mathfrak{q}_{\text{aceita}}}\right], \left[\frac{\gamma_r}{\gamma_r}\right], \left[\frac{\#}{\#}\right], \ldots,$$
$$\left[\frac{\#}{\#}\right], \left[\frac{\gamma_1}{\gamma_1}\right], \left[\frac{\gamma_2\mathfrak{q}_{\text{aceita}}}{\mathfrak{q}_{\text{aceita}}}\right], \left[\frac{\gamma_r}{\gamma_r}\right], \left[\frac{\#}{\#}\right],$$
$$\left[\frac{\gamma_1\mathfrak{q}_{\text{aceita}}}{\mathfrak{q}_{\text{aceita}}}\right], \left[\frac{\gamma_r}{\gamma_r}\right], \left[\frac{\#}{\#}\right], \left[\frac{\mathfrak{q}_{\text{aceita}}\gamma_r}{\mathfrak{q}_{\text{aceita}}}\right], \left[\frac{\#}{\#}\right], \left[\frac{\mathfrak{q}_{\text{aceita}}\#\#}{\#}\right]$$

Fonte: Autor (2024)

Vemos que estes dominós estão em quantidade finita e, quando há a história de computação de aceitação, o emparelhamento com os dominós que dispomos

acontece.

E isto ocorre mesmo quando o estado inicial é o próprio estado de aceitação. Pois, sendo $\mathfrak{q}_0 = \mathfrak{q}_{\text{aceita}}$, à direita do dominó inicial $\left[\frac{\#}{\#\mathfrak{q}_0 w_1 w_2 \ldots w_k \#}\right]$ colocaremos o dominó $\left[\frac{\mathfrak{q}_0 w_1}{w_1}\right]$, o que iniciará com os pseudopassos a completarem o emparelhamento também.

Assim, quando $\mathcal{M}$ aceita w, ocorre que $g\left(\langle \mathcal{M}, w\rangle\right) = P' \in$ PCPM.

Parte B. 2. Agora mostraremos que todos os emparelhamentos possíveis dessas instâncias do PCPM que são imagem de g incluem sempre, desde seu início, uma das possibilidades listadas a seguir.

Ou inclui o emparelhamento visto acima, baseado na história de computação de aceitação. Ou inclui uma variação dele. Tal variação se dá com relação aos pseudopassos, aqueles após a configuração completa de aceitação.

Ou seja, outros emparelhamentos existentes sempre trarão, em seu início, algum emparelhamento que tenha toda a história de computação de aceitação em questão. Podem ser os mais simples dentre eles, apenas acrescidos de mais dominós à frente. Poderíamos colocar, por exemplo, $\left[\frac{\#}{\#}\right]$ à frente, ou dominós do formato $\left[\frac{\beta_1}{\beta_1}\right]$. Ou mesmo repetir a sequência feita para o emparelhamento primário após ele mesmo.

Quanto às variações, elas se dão no sentido de que, dependendo da posição da cabeça de R$(\mathcal{M})$ nos seus quadrados, podemos optar por ordenar os dominós que incluem o estado de aceitação em cima de mais de um jeito. No exemplo visual que demos logo acima, optamos por ir colocando os dominós que cobriam desde $\gamma_{r-1}\mathfrak{q}_{\text{aceita}}$, indo em ordem decrescente até $\gamma_1 \mathfrak{q}_{\text{aceita}}$ e só depois colocamos o dominó com $\mathfrak{q}_{\text{aceita}}\gamma_r$.

Mas seria possível colocar em outras ordens, como iniciando em $\gamma_{r-1}\mathfrak{q}_{\text{aceita}}$, depois colocando $\mathfrak{q}_{\text{aceita}}\gamma_r$, e daí decrescermos desde de $\gamma_{r-2}\mathfrak{q}_{\text{aceita}}$ até $\gamma_1\mathfrak{q}_{\text{aceita}}$. Como também poderia ser com outras ordens.

Além de que, conseguiríamos, em quantidade limitada, colocarmos dominós $\left[\frac{\#}{\sqcup\#}\right]$ no lugar de dominós $\left[\frac{\#}{\#}\right]$ dentro dos pseudopassos, alongando o seu número.

Em verdade, um mesmo elemento de PCPM pode ser imagem de mais de um elemento de MM. Basta que duas máquinas de Turing distintas tenham todos os seus elementos da 7-upla identicamente codificados, exceto o alfabeto de entrada, sendo que o alfabeto de entrada da primeira é subconjunto do alfabeto de entrada da segunda.

Podemos ver que, na construção dos dominós não é feita distinção entre os alfabetos. Então, para uma entrada que for aceita na primeira máquina acima citada, a instância P' baseada nessa dupla será a mesma instância que seria baseada na

dupla da mesma entrada, com a segunda máquina.

Uma outra possibilidade é de as codificações de duas máquinas de Turing distintas difiram entre si apenas nas transições feitas a partir das configurações cujo estado é um estado de parada. Afinal, estas configurações nunca darão origem às suas imagens de fato na computação. Também neste caso, a instância P' é a mesma.

E claro, também podemos combinar estes dois fatores, desta diferença apontada com relação ao alfabeto de entrada e das transições a partir dos estados de parada. Duas máquinas de Turing que difiram em suas configurações somente por causa destes pontos darão origem à mesma instância P'.

Mas, fora isso, não há outra situação de mesma imagem através de g. Pois nesta função, funções de transição com diferentes comportamentos modificam os dominós. Alfabetos da fita de diferentes tamanhos também. O mesmo para o conjunto de estados. E a escolha de qual será o estado inicial, do estado de aceitação e do estado de rejeição, cada uma, resultam também em diferentes conjuntos de dominós.

Dito isso, se houver emparelhamento possível, dividiremos a tarefa agora em duas. Trata-se de **i)** mostrar que não podemos interromper um emparelhamento tal como mostrado antes a fim de reiniciar a história de computação, intercalando duas histórias ou mais. E **ii)** mostrar que o emparelhamento visto (baseado na história de computação de aceitação) ou uma de suas variações serão incluídos necessariamente.

Parte B. 2. i. Algo imaginável de ser incluído para que ocorresse um emparelhamento, mas que em verdade é impossível de sê-lo, se trata de uma intercalação de configurações completas, em ordem, a partir de um dado momento. Estendendo-se até um outro momento, tais configurações completas seriam advindas da mesma história de computação de aceitação, colocada repetidas vezes.

Quanto às intercalações, se existissem, estariam baseadas na colocação de outro dominó $\left[\frac{t_1}{s_1}\right]$ no meio de uma história de computação. Suponhamos que exista $H = (C_1, C_2, \ldots, C_l)$, a história de computação de aceitação de R($\mathcal{M}$) sobre w, com l sendo o número natural da configuração completa de aceitação. Após a primeira vez que H começar a ser montada com os dominós, hipoteticamente poderíamos, no lugar de um dominó $\left[\frac{\#}{\#}\right]$, colocar o dominó $\left[\frac{t_1}{s_1}\right]$, pois ele também começa com $\#$ em suas duas cadeias. Mas vamos mostrar que isto não produz qualquer emparelhamento.

Sendo um natural l_1, com $1 < l_1 < l$, o dominó inicial pode ser colocado, olhando para a cadeia de baixo, logo após a configuração completa C_{l_1}, ou após alguma das vezes em que se monta algo da fita de R($\mathcal{M}$) que tem o estado de aceitação, o que chamamos antes de pseudopassos.

Após isso, para completar o emparelhamento, seria necessário então revezar,

a partir deste segundo dominó $\left[\frac{t_1}{s_1}\right]$, para que os alinhamentos sejam montados ora com as configurações completas ou com os pseudopassos da primeira H, ora com as configurações completas ou com os pseudopassos da segunda H.

Compreendamos que os pseudopassos são considerados, embaixo, como sendo a reprodução da configuração completa de aceitação de H sempre com um símbolo a menos. E, seja r a quantidade de símbolos de Γ representados na configuração completa C_l montada nos dominós. Então haverá, para cada uma das duas repetições de H, um número de r_1 e de r_2 pseudopassos, com $r_1, r_2 \geq r$ (por causa da possibilidade de se colocar dominós $\left[\frac{\#}{\sqcup\#}\right]$).

Chamaremos, então, cada um deles, pela ordem em que aparecem ao longo do pretenso emparelhamento, de $F_{1,1}, F_{2,1}, \ldots, F_{r_1,1}$, para a primeira vez que terminamos H. E, da mesma forma, chamaremos de $F_{1,2}, F_{2,2}, \ldots, F_{r_2,2}$ os pseudopassos para a segunda vez que terminamos H. Assim, para cada H montada, ela terminaria com seus pseudopassos mais o dominó $\left[\frac{\mathfrak{q}_{\text{aceita}}\#\#}{\#}\right] = \left[\frac{F_{r_1,1}\#\#}{\#}\right] = \left[\frac{F_{r_2,2}\#\#}{\#}\right]$.

Vamos também considerar o primeiro dos dominós como $\left[\frac{t_1}{s_1}\right] = \left[\frac{\#}{\#C_1\#}\right]$. Mais a seguir, na figura 112, uma ilustração simplificada do que estamos falando, onde o primeiro dominó foi colocado apenas mais uma segunda vez, o sendo após a configuração C_{l_1}, com $l_1 > r + 1$ neste exemplo. Para isso, tomaremos $i = l + r_1 - l_1$, usado na ilustração.

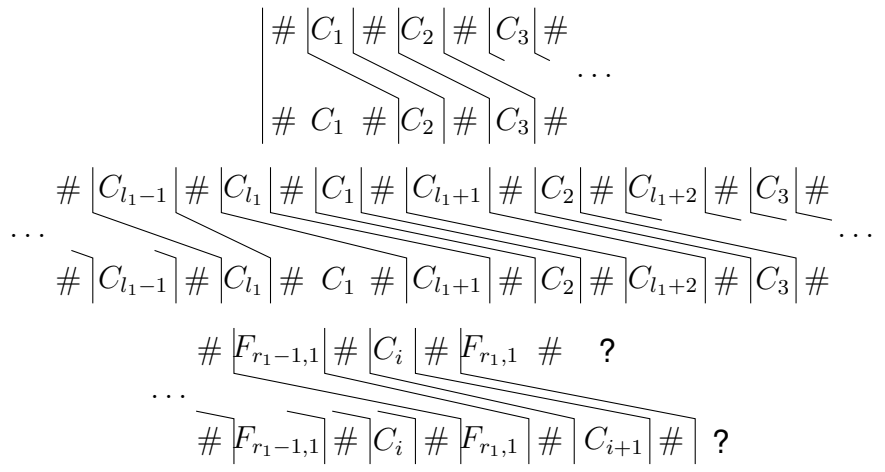
Nós temos a mesma históra H sendo reproduzida duas vezes. Quando ela o era uma só vez, o dominó $\left[\frac{\mathfrak{q}_{\text{aceita}}\#\#}{\#}\right]$ terminava seus pseudopassos seguintes. Contudo, agora as configurações completas e pseudopassos da primeira reprodução intercalam-se com os da segunda.

Originalmente conseguíamos colocar o dominó $\left[\frac{F_{r_1,1}\#\#}{\#}\right]$ no fim, pois a diferença de comprimento entre as cadeias de cima de debaixo era de apenas duas unidades. Para que a cadeia de cima alcançasse o emparelhamento com a debaixo, lhe faltava apenas $\mathfrak{q}_{\text{aceita}}$ e $\#$, respectivamente. Mas com a repetição de H, não conseguimos fazer o mesmo.

A diferença de comprimento já é maior do que duas unidades, e não é possível finalizar apenas a primeira reprodução e continuar com a segunda, pois embora a cadeia de cima deva receber exatamente um $\mathfrak{q}_{\text{aceita}}$ e um $\#$, ela não pode receber o segundo $\#$ que há no mesmo dominó, pois o que há na continuação da cadeia de baixo já é a próxima configuração ou o próximo pseudopasso da segunda reprodução de H.

Ao mesmo tempo, não há dominó que possua apenas $\mathfrak{q}_{\text{aceita}}$ em sua cadeia de cima, como também não há dominó, além de $\left[\frac{\mathfrak{q}_{\text{aceita}}\#\#}{\#}\right]$, que comece com $\mathfrak{q}_{\text{aceita}}\#$

Figura 112 – Desenho da tentativa de intercalação para um emparelhamento com os nossos dominós



Fonte: Autor (2024)

em cima. Portanto, não há como prosseguir, daí. Não haverá emparelhamento.

Vale observar, também, que o primeiro dominó só conseguiria ser uma outra vez colocado no lugar de um dominó que poderia ser $\left[\frac{\#}{\#}\right]$, mas não $\left[\frac{\#}{\sqcup\#}\right]$. Então nem sequer é no fim de qualquer configuração completa que se poderia, hipoteticamente, acrescentar novamente o primeiro dominó.

E ainda, a impossibilidade de emparelhamento não seria diferente mesmo se acrescentássemos o dominó $\left[\frac{t_1}{s_1}\right]$ uma terceira vez, estendendo a intercalação. Haveria alternância entre três reproduções de H, cada uma começando em momentos diferentes, mas cairíamos no mesmo problema de finalizar a primeira delas em algum momento.

Analogamente se acrescentássemos $\left[\frac{t_1}{s_1}\right]$ uma quarta vez, uma quinta, enfim, quantas vezes, de maneira finita, quiséssemos. Não há como fazê-lo e chegar a um emparelhamento. E com isso, já deixamos provado que, após uma configuração completa, embaixo, os únicos dominós que a podem suceder imediatamente são ou $\left[\frac{\#}{\#}\right]$, ou $\left[\frac{\#}{\sqcup\#}\right]$.

Parte B. 2. ii. Agora, mostraremos que só há os emparelhamentos conforme PCPM se há a história de computação de aceitação de R$(\mathcal{M})$ sobre w.

Pois bem. Após o obrigatório dominó $\left[\frac{t_1}{s_1}\right]$ inicial, há os casos em que $k > 0$ e em que $k = 0$. Em ambos, o próximo dominó tem que ser aquele cujo primeiro símbolo acima é $\mathfrak{q}_0$. Só que os únicos dominós que temos neste formato são aqueles destinados à transição ou, no caso em que o estado inicial é o estado de aceitação, os dominós $\left[\frac{\mathfrak{q}_0\beta_1}{\mathfrak{q}_0}\right]$. Tratemos desse segundo caso depois.

Sabemos, como é o início da fita de R$(\mathcal{M})$, que o movimento será para a direita.

Consequentemente teremos, acima no dominó, a configuração inicial, e abaixo, o símbolo impresso e o estado sucessor, respectivamente. E este dominó é o único que traz tal configuração acima, e embaixo ele continua exatamente com o que a configuração completa faria.

Depois podemos ter ou não o fim da fita de R($\mathcal{M}$). Se não, então para completar com os símbolos da cadeia da primeira configuração completa, só há dominós no formato $\left[\frac{\beta_1}{\beta_1}\right]$, pois outros dominós com símbolos de Γ em cima contêm estados. E assim se vai preenchendo. Como já mostramos, depois de uma configuração completa, deveremos colocar ou $\left[\frac{\#}{\#}\right]$, ou $\left[\frac{\#}{\sqcup\#}\right]$.

Em todos os dominós de até então na sequência, a cadeia de baixo neles era maior ou igual, em comprimento, à cadeia de cima. Também é verdade que, se o estado da configuração completa, na cadeia de baixo, já chega ao fim da fita, (como seria com o caso da cadeia de entrada vazia ou com uma cadeia de entrada unitária), o dominó $\left[\frac{\#}{\sqcup\#}\right]$ é obrigatório a seguir.

Afinal, para que mais à frente algum dominó alinhe, em cima, o referido estado que já está ali embaixo no fim da fita, tal dominó precisa ter um símbolo de Γ imediatamente após tal estado. Lembremos que não há dominós com estados de Q no fim de sua cadeia particular debaixo, exceto pelo estado $\mathfrak{q}_{\text{aceita}}$, do qual não é o caso neste momento. Como também não há dominós com um estado de Q seguido de $\#$ no fim de sua cadeia debaixo.

E vemos que o padrão descrito nos parágrafos acima se repetirá. O que pode mudar é apenas se a próxima configuração vai exigir uma transição para a esquerda ou para a direita.

No caso da esquerda, o dominó traz um símbolo antes do estado, que pode ser qualquer um de Γ. Mas mesmo assim, ele só vai se encaixar se aquele símbolo já estava na cadeia debaixo do pretenso emparelhamento, e se a configuração à frente de tal símbolo acima é a configuração que já estava na configuração completa abaixo. E em qualquer um dos casos, de direita e de esquerda, o comprimento das cadeias dos dominós ainda seguirá o mesmo ditame anterior.

Claramente a sequência de dominós seguirá esse padrão até chegar ao estado de aceitação ou ao de rejeição. Se não chegar a nenhum dos dois, não haverá emparelhamento, visto que a cadeia debaixo continuará maior em comprimento que a cadeia de cima.

Se a sequência chega ao estado de rejeição também não haverá emparelhamento, pois não formamos nenhum dominó que possui o estado de rejeição na sua cadeia de cima, que se alinharia com o estado de rejeição que já há embaixo, sucessor da última configuração.

Se a sequência chega ao estado de aceitação, o emparelhamento que mostra-

mos acontece, ou variações dele, como comentadas. E alguma delas deve acontecer para haver emparelhamento. Afinal, a partir do momento em que fecha a configuração completa de aceitação, é necessário diminuir a diferença de tamanho entre a cadeia de cima e a de baixo.

Isso não é possível apenas com dominós puramente de símbolos de Γ, como já vimos. Deve acontecer, então, com dominós que incluam o estado de aceitação em cima. E os dominós que temos assim são aqueles dos formatos $\left[\frac{\beta_1 \mathfrak{q}_{\text{aceita}}}{\mathfrak{q}_{\text{aceita}}}\right]$ e $\left[\frac{\mathfrak{q}_{\text{aceita}} \beta_1}{\mathfrak{q}_{\text{aceita}}}\right]$. Sendo assim, temos que os usar aqui, e respeitando os símbolos que, na cadeia debaixo, já estavam ao lado do estado de aceitação. Assim, com estes dominós e com os dominós de formato $\left[\frac{\beta_1}{\beta_1}\right]$ e $\left[\frac{\#}{\#}\right]$, vamos repetindo, ao final, a última configuração completa, mas com cada vez menos símbolos nela, até que sobre ali, na última repetição, apenas o estado de aceitação.

E estes últimos passos, com o dominó de aceitação e os seguintes, valem também para o caso que falamos no início. Isto é, para quando o estado inicial é o estado de aceitação. Já começaremos tendo que continuar a sequência de dominós com o dominó $\left[\frac{\mathfrak{q}_0 \beta_1}{\mathfrak{q}_0}\right]$, visto que, sendo estado de parada, $\mathfrak{q}_0$ não aparecerá dentre os dominós de transição.

E com ele, seguem os dominós com o restante da configuração completa e os pseudopassos. Tanto para este caso particular (de $\mathfrak{q}_0 = \mathfrak{q}_{\text{aceita}}$), como para os outros casos, vale o que seguiremos falando.

Até podemos, neste ponto (dos pseudopassos), no lugar de $\left[\frac{\#}{\#}\right]$, colocar dominós $\left[\frac{\#}{\sqcup\#}\right]$ um número finito de vezes. Mas se trocarmos o dominó anterior por este último de um modo ilimitado, nunca chegaremos ao emparelhamento pressuposto. Isso porque, embora a cadeia de cima, a cada pseudopasso, diminuísse em uma unidade a diferença dos comprimentos, este último dominó anula essa diminuição ao acrescentar um $\sqcup$ ao lado dos últimos símbolos abaixo antes do próximo $\#$. Sendo assim, a diferença entre o comprimento das cadeias ainda deve se encaminhar para a mesma conclusão.

Pois que, após todos esses passos, a diferença de comprimento entre as cadeias será, tal como no exemplo, de apenas duas unidades. E a anulamos com o dominó $\left[\frac{\mathfrak{q}_{\text{aceita}} \# \#}{\#}\right]$. Este, aliás, era o único dominó que poderia vir agora. Pois os últimos símbolos da cadeia debaixo são $\mathfrak{q}_{\text{aceita}}$ e $\#$, não há dominós com apenas $\mathfrak{q}_{\text{aceita}}$ em cima, e ele é o único dominó com $\#$ à direita de $\mathfrak{q}_{\text{aceita}}$. E, deste modo, ele completa o emparelhamento primordial.

Assim, todo emparelhamento que vier, deve começar com este emparelhamento primordial ou com uma variação dele. E tal emparelhamento nos mostra que

existe a história de computação de aceitação de R($\mathcal{M}$) sobre w e, consequentemente, de $\mathcal{M}$ sobre w.

Parte C. Comecemos tomando a função $f : \Sigma_{\mathsf{e}}^* \to \Sigma_{\mathsf{e}}^*$ tal que, se $x \in MM$, então $f(x) = g(x)$, e se $x \notin MM$, então $f(x) = \varepsilon$.

É visível que, seguindo padrão similar ao da codificação das máquinas de Turing, os dominós seguidos do dominó escolhido para primeiro não precisam de muito mais representações em Σ_{e} que aquelas já usadas em vista das máquinas de Turing para representá-los. Usa-se algo para o $\#$, algo para separar a cadeia de cima da cadeia de baixo, e algo para delimitar o início e fim de um dominó.

Dito isso, vamos fazer o seguinte. Construamos a máquina de Turing descrita abaixo, segundo a codificação que estamos utilizando.

$\mathcal{M}' =$ **“** Sobre a entrada x:

1. Verifique se ela é uma entrada $\langle \mathcal{M}, w \rangle \in MM$. Se sim, passe para o próximo passo. Se não, descarte.

2. Construa a máquina R($\mathcal{M}$).

3. A partir de $\langle$R($\mathcal{M}$)$, w \rangle$, construa, como saída, a instância P' como antes descrita, e então aceite. **”**

Ora, para construir os dominós da instância é necessário verificar:

- a cadeia de entrada w, no caso do dominó $\left[\frac{t_1}{s_1}\right]$;
- cada símbolo, no caso dos dominós $\left[\frac{\beta_1}{\beta_1}\right]$, $\left[\frac{\beta_1 \mathfrak{q}_{\text{aceita}}}{\mathfrak{q}_{\text{aceita}}}\right]$ e $\left[\frac{\mathfrak{q}_{\text{aceita}} \beta_1}{\mathfrak{q}_{\text{aceita}}}\right]$;
- as transições, no caso dos dominós de transição para a direita;
- as transições e cada símbolo, no caso dos dominós de transição para a esquerda;
- nada, no caso dos dominós $\left[\frac{\#}{\#}\right]$ e $\left[\frac{\#}{\textvisiblespace\#}\right]$;
- a posição dos estados de parada no caso de $\left[\frac{t_1}{s_1}\right]$, $\left[\frac{\beta_1 \mathfrak{q}_{\text{aceita}}}{\mathfrak{q}_{\text{aceita}}}\right]$ e $\left[\frac{\mathfrak{q}_{\text{aceita}} \beta_1}{\mathfrak{q}_{\text{aceita}}}\right]$ e $\left[\frac{\mathfrak{q}_{\text{aceita}} \# \#}{\#}\right]$.

Todas estas coisas são acessíveis a partir da entrada $\langle \mathsf{R}(\mathcal{M}), w\rangle$. Claramente, uma máquina de Turing é capaz de fazer esse trabalho de verificação e construção.

E também é capaz de destacar o dominó que definimos como $\left[\frac{t_1}{s_1}\right]$ de modo que seja o segundo componente do par ordenado P'. Ela verificará que o dominó ao qual cabe este papel é o único dominó em P' que possui $\#$ seguido de $\mathfrak{q}_0$, na cadeia de baixo, se tratando justamente de $\left[\frac{t_1}{s_1}\right]$.

Falamos, no último item listado, de verificar a posição dos estados de parada por causa do caso em que $\mathfrak{q}_0 = \mathfrak{q}_{\mathsf{aceita}}$. Se visto que são iguais, a máquina terá de codificar os dominós relacionados com os mesmos símbolos no que diz respeito ao estado neles representado.

Refletindo as entradas e saídas desta máquina, temos justamente a função $f : \Sigma_{\mathsf{e}}^* \to \Sigma_{\mathsf{e}}^*$, tal que $f(x) = \varepsilon$ se $x \neq \langle \mathcal{M}, w\rangle \in MM$, e $f(x) = g(x) = P'$ se $x = \langle \mathcal{M}, w\rangle \in MM$.

Vemos que a máquina sempre para sobre uma saída, sendo esta saída $f(x) \in \Sigma_{\mathsf{e}}^*$ a cadeia vazia ou uma saída P'.

Ela é a cadeia vazia se, e somente se a entrada x não se refere a uma máquina $\mathcal{M}$ com uma cadeia w adequada.

Como já visto ao longo dos itens **A)** e **B)**, através da função g, a instância P' que surge a partir de uma dupla de $\mathcal{M}$ e w terá o emparelhamento requisitado para PCPM se, e somente se $\mathcal{M}$ aceita w, isto é, se e somente se $\langle \mathcal{M}, w\rangle \in \mathsf{A}_{\mathsf{MT}}$.

Assim, com tudo isto, é verdade, para todo $x \in \Sigma_{\mathsf{e}}^*$, que $x \in \mathsf{A}_{\mathsf{MT}} \iff f(x) \in$ PCPM. Portanto, também é verdade que $\mathsf{A}_{\mathsf{MT}} \leq_{\mathsf{m}}$ PCPM. ■

E agora, o último lema que precede o teorema com o qual concluiremos o assunto do capítulo. Veremos que, para cada uma das instâncias P' do problema da correspondência de Post modificado tem uma instância correspondente P do problema original. Ocorre que havíamos colocado como uma imposição o fato de que, para ser elemento de PCPM, a codificação da instância deveria dizer respeito a um conjunto de dominós cujo emparelhamento começasse por um dominó $\left[\frac{t}{s}\right]$ do conjunto.

No entanto, se dependesse apenas do problema original, todas as instâncias montadas como na demonstração acima, a partir de uma máquina de Turing e de uma entrada, teriam um emparelhamento e não apenas aquelas que surgem de um elemento de A_{MT}. Pois basta tomarmos dominós seus do formato $\left[\frac{\beta_1}{\beta_1}\right]$ ou simplesmente um dominó $\left[\frac{\#}{\#}\right]$. Por si só eles já concluem um emparelhamento.

Só que os emparelhamentos de que vínhamos tratando nos itens anteriores estavam correlacionados à história de computação de aceitação de $\mathsf{R}(\mathcal{M})$ sobre w, começando com $\left[\frac{t_1}{s_1}\right]$. O que faremos, agora, é converter instâncias P' quaisquer,

relativas a o PCPM, cada qual em uma instância correspondente P, relativa ao PCP, cujo emparelhamento depende do emparelhamento de P' no PCPM.

E, basicamente, montaremos, a partir de P' cuja segunda componente fosse $\left[\frac{t_1}{s_1}\right]$, uma instância P cujos únicos emparelhamentos possíveis precisam iniciar por um dominó associado a $\left[\frac{t_1}{s_1}\right]$.

Lema 10.2.2. *É verdade que PCPM $\leq_m$ PCP.*

Demonstração. Sendo $a \in \mathbb{N} - \{0\}$ um número qualquer, seja $u = u_1 u_2 \dots u_a$ uma cadeia qualquer de comprimento a, e $*$ um símbolo que não pertença a $\Gamma \cup \{\#\}$. Definimos as cadeias $\star u$, $u\star$ e $\star u\star$ como

$$\star u = *u_1 * u_2 * u_3 * \cdots * u_a,$$

$$u\star = u_1 * u_2 * u_3 * \cdots * u_a*,$$

e

$$\star u\star = *u_1 * u_2 * u_3 * \cdots * u_a * .$$

Assim, a cadeia $\star u$ é resultado de se colocar um símbolo $*$ antes de cada símbolo de u. A cadeia $u\star$ o é de se colocar um símbolo $*$ depois de cada símbolo de u. E $\star u\star$, de se tomar u e deixá-la com exatamente um $*$ antes e depois de cada um de seus símbolos.

Deste modo, com um $p \in \mathbb{N} - \{0\}$, se a nossa instância P', relativa ao PCPM, era

$$P' = \left\langle \left\{ \left[\frac{t_1}{s_1}\right], \left[\frac{t_2}{s_2}\right], \left[\frac{t_3}{s_3}\right], \dots, \left[\frac{t_p}{s_p}\right] \right\}, \left[\frac{t_1}{s_1}\right] \right\rangle,$$

construiremos a instância P, relativa ao PCP, tal que

$$P = \left\{ \left[\frac{\star t_1}{\star s_1 \star}\right], \left[\frac{\star t_1}{s_1 \star}\right], \left[\frac{\star t_2}{s_2 \star}\right], \left[\frac{\star t_3}{s_3 \star}\right], \dots, \left[\frac{\star t_p}{s_p \star}\right], \left[\frac{*\lozenge}{\lozenge}\right] \right\}.$$

Aqui, $\lozenge$ é um símbolo usado para esta instância, cujo dominó será aquele que finaliza o emparelhamento primordial. O que acontecerá aqui é que temos uma instância do PCP, mas para que possa haver emparelhamento, este precisa começar pelo dominó $\left[\frac{\star t_1}{\star s_1 \star}\right]$. Afinal, este é o único dominó cujas cadeias de cima e debaixo iniciam pelo mesmo símbolo, o qual se trata justamente de $*$.

Depois disso, o pretenso emparelhamento da instância P segue basicamente o mesmo raciocínio que o da instância P' seguia. Os símbolos $*$ entre os símbolos das cadeias dos dominós não vão interferir nisso.

Vemos que todos os dominós, exceto um, não são capazes de serem o último do emparelhamento, pois eles terminam na cadeia de baixo com o símbolo $*$ e na

cadeia de cima com algum símbolo distinto de $*$. E, para que o emparelhamento seja fechado, utilizamos exatamente o dominó $\left[\frac{*\Diamond}{\Diamond}\right]$.

É perceptível que a partir de P' conseguimos construir P com uma máquina de Turing. A partir de cada dominó que era de P', a máquina formará um novo dominó colocando um $*$ à esquerda de cada símbolo nas cadeias de cima, e um $*$ à direita de cada símbolo nas cadeias de baixo. E não esquecendo da formação do dominó $\left[\frac{*\Diamond}{\Diamond}\right]$.

E, indicado pelo segundo componente do par ordenado P', a máquina de Turing terá a informação sobre a partir de qual dominó fazer também a versão com mais uma estrela.

Ora, se P' tinha o emparelhamento requisitado por PCPM, será semelhante em P, com seus dominós. O emparelhamento de P deverá incluir, em seu início, alguma das sequências dispostas dos dominós correspondentes aos dominós do emparelhamento requisitado por PCPM que havia em P'.

Se existia emparelhamento sob a condição do PCPM em P', após serem sequenciados os dominós correspondentes de P, imediatamente após eles o novo emparelhamento será fechado pelo dominó $\left[\frac{*\Diamond}{\Diamond}\right]$. Pois, diferente de em P', depois de aquele correspondente ao antigo último dominó de um emparelhamento ser colocado, ainda restará uma diferença de comprimento entre a cadeia de cima e a cadeia de baixo. Diferença de uma unidade, onde falta um símbolo $*$ ao final da cadeia de cima.

O emparelhamento só ocorrerá quando, após estes dominós, vier $\left[\frac{*\Diamond}{\Diamond}\right]$. E, claramente, isso não afeta em nada o fato de que os dominós de P chegaram a um emparelhamento baseado no emparelhamento de P'.

E ainda, depois de colocado o dominó $\left[\frac{*\Diamond}{\Diamond}\right]$, poderíamos colocar o dominó $\left[\frac{\star t_1}{\star s_1 \star}\right]$, que é o único que começa com o mesmo símbolo em cima e embaixo. E partiríamos para outros emparelhamentos possíveis, mais longos que anteriores. E essa continuidade apenas repetiria outra sequência de dominós baseada em algum emparelhamento de P', conforme requisitado pelo PCPM.

Em verdade, isso se dá com todos os emparelhamento que quisermos formar colocando dominós $\left[\frac{*\Diamond}{\Diamond}\right]$ em quantidade finita. Entre dois dominós $\left[\frac{*\Diamond}{\Diamond}\right]$, o que haverá é uma sequência de dominós de P baseada em uma sequência de dominós de P' que faz um emparelhamento.

E é claro que, se P' não tinha o emparelhamento requisitado pelo PCPM, então P não terá emparelhamento. Tal pretenso emparelhamento só poderia começar por $\left[\frac{\star t_1}{\star s_1 \star}\right]$. Porém, seguiria a sequência correspondente à dos dominós de P'.

Afinal, a cadeia sequenciada nos dominós de P, em cima ou embaixo, seria basicamente aquela onde os elementos das posições ímpares são $*$ e os elementos

das posições pares são os elementos da cadeia que haveria a partir da sequência dos dominós de P'. Os símbolos $*$ não trazem nada de novo nessa questão do emparelhamento então.

O único dominó de P que talvez interferisse nisso, por sua construção não necessitar de algum dominó de P', é o $\left[\frac{*\Diamond}{\Diamond}\right]$. Entretanto, ele nunca poderá ser colocado porque, para isso, precisa que a diferença entre os comprimentos das cadeias seja de uma unidade. Ele é o único dominó com o símbolo $\Diamond$. E a diferença entre os comprimentos é sempre maior que um. Então ele não muda essa conjuntura.

Sendo assim, construamos, abaixo, a máquina de Turing de que falamos antes.

$\mathcal{M}$ = " Sobre a entrada x:

1. Verifique se a entrada x é igual a P', codificação de uma instância do problema da correspondência de Post modificado, isto é, sendo o par ordenado composto por, respectivamente, um conjunto de dominós e um de seus dominós. Se sim, passe para o próximo passo. Se não, descarte.

2. A partir de P', construa a saída $\langle P \rangle$, como antes foi descrita. E então aceite. "

Com a máquina, definimos a função $f : \Sigma_{\mathsf{e}}^* \to \Sigma_{\mathsf{e}}^*$, tal que $f(x) = \varepsilon$ se $x \neq P'$, e $f(P') = \langle P \rangle$, onde $P' \in$ PCPM.

Bom, se $x \neq P'$, então sua imagem através de f não é sequer um elemento de PP. Se $x = P'$ e P' não tem emparelhamento sob a condição de PCPM, então $f(x) = \langle P \rangle \notin$ PCP, pois, como expúnhamos antes, o emparelhamento de P depende do emparelhamento de P'. E, por conta disso, se $x = P'$ e P' tem tal emparelhamento requisitado, então $f(x) = \langle P \rangle \in$ PCP.

Assim, é verdade, para todo $x \in \Sigma_{\mathsf{e}}^*$, que $x \in$ PCPM $\iff f(x) \in$ PCP. E portanto, é verdade que PCPM $\leq_{\mathsf{m}}$ PCP. ■

E, finalmente, o nosso teorema.

Teorema 10.2.3. *A linguagem PCP é indecidível.*

Demonstração. A partir dos lemas 10.2.1 e 10.2.2 acima, da proposição 7.2.7 (sobre a transitividade de $\leq_{\mathsf{m}}$) e do corolário 7.2.3 (sobre a indecidibilidade de A para B se $A \leq_{\mathsf{m}} B$), concluímos o que o enunciado deste teorema está correto. ■

11 TEOREMA DA RECURSÃO

Este é o último capítulo antes das considerações finais. Veremos um resultado pelo qual tomamos conhecimento de que há máquinas que conseguem imprimir a descrição de si mesmas e, além disso, fazem outras operações que quisermos. É o chamado **teorema da recursão**. Para que cheguemos até sua conclusão, usaremos alguns resultados e definições auxiliares.

Definição 20. Sendo Σ um alfabeto qualquer, para toda entrada $w \in \Sigma^*$, definimos a máquina $\mathcal{P}_w$ que, para qualquer entrada, imprime w no início da fita, volta para o primeiro quadrado e para.

Não é difícil ver que tal máquina existe. Podemos montá-la de modo a ir até o fim da entrada que receber sem modificá-la e, ao chegar no primeiro $\sqcup$, volta até o início apagando cada símbolo, parando quando escanenar o primeiro $\sqcup$ (sinal de que já está no primeiro quadrado).

A partir daí, ela imprime a cadeia w. Para isto, podemos usar $|w|$ estados distintos para imprimi-la. Pois se $w = \varepsilon$, já a temos, sem necessitar de mais nada por agora. E se $|w| > 0$, então cada um destes estados imprimirá um dos símbolos da cadeia, da esquerda para a direita.

E, como vimos na proposição 7.2.1, é possível termos uma máquina que faz tudo isso e que volta para o primeiro quadrado ao fim de sua atividade.

Dito isto, vamos para um resultado a respeito da máquina $\mathcal{P}_w$.

Lema 11.0.1. *Existe uma função computável* $q : \Sigma^* \to \Sigma^*$ *tal que,* $q(w) = \langle \mathcal{P}_w \rangle$.

Demonstração. Seja Σ_1 o alfabeto do qual queremos retirar nossas entradas e Σ_2 o alfabeto que fornece as entradas para codificações de máquinas de Turing quaisquer. Tomemos $\Sigma = \Sigma_1 \cup \Sigma_2$. Vamos definir a função $q : \Sigma^* \to \Sigma^*$ tal que $q(w) = \langle \mathcal{P}_w \rangle$.

Seguindo os passos anteriormente descritos nesta seção, a partir da entrada w, podemos montar a máquina $\mathcal{Q}$ descrita abaixo, cujo alfabeto de entrada é Σ.

$\mathcal{Q} =$ **“** Sobre a entrada w:

1. Codifique w sob o alfabeto Σ_2.
2. A partir de $\langle w \rangle$, construa a máquina de Turing $\mathcal{P}_w$, isto é, a máquina a seguir.

   $\mathcal{P}_w =$ **“** Sobre qualquer entrada:

   1. Apague a entrada e volte para o começo.
   2. Escreva w na fita.

3. Volte para o começo e então aceite. **”**

3. Forneça a cadeia $\langle \mathcal{P}_w \rangle$ como saída e então aceite.**”**

O primeiro passo pode se dar tendo algum estado de $\mathcal{Q}$ o qual, dependendo do símbolo escaneável de Σ, leve a computação a uma codificação específica a ser desdobrada. Como o símbolo que é o α_1 levar à impressão da codificação DC, o símbolo α_3 levar à impressão da codificação DCCC, o α_7 à impressão de DCCCCCCC, etc.

Agora, a respeito dos próximos passos, sabemos como proceder quanto a descrever os dois primeiros passos para a codificação de $\mathcal{P}_w$. Descrevemo-los logo após a definição 20. E quanto ao terceiro, também sabemos, pois o descrevemos na observação 9 (seção 7.2), sobre fornecer, por meio de uma máquina de Turing, a descrição de uma outra máquina que faça o trabalho e volte ao começo.

Com a máquina $\mathcal{Q}$, fica satisfeita a prova de que q é uma função computável. ■

Agora, com a função q, iremos construir uma máquina cujo papel é o de dar a descrição de si mesma. Para isso ela é dividida em duas partes, $\mathcal{A}$ e $\mathcal{B}$, para que consiga fazer referência a si mesma. Assim, o resultado que buscamos é a máquina que combina $\mathcal{A}$ e $\mathcal{B}$, a qual, sobre qualquer entrada, dá como saída $\langle \mathcal{A}\,\mathcal{B} \rangle$.

Em vista disso, a parte $\mathcal{A}$ vai imprimir a descrição de $\mathcal{B}$. Usará como subrotina a máquina $\mathcal{P}_{\langle \mathcal{B} \rangle}$. O que fará a máquina $\mathcal{B}$? Imprimirá ainda a descrição de $\mathcal{A}$ usando $\mathcal{P}_{\langle \mathcal{A} \rangle}$? Se assim fosse, teríamos a parte $\mathcal{A}$ em função de $\mathcal{B}$ e a $\mathcal{B}$ em função de $\mathcal{A}$. Só que isso não pode acontecer, pois estaríamos violando a lógica, basicamente com uma petição de princípio, onde queremos chegar a uma conclusão e agimos como se já tivéssemos posse dela.

Para fazer aquilo a que nos propomos, vamos definir como seria $\mathcal{B}$ sem ser em função de $\mathcal{A}$. Enquanto a parte $\mathcal{A}$ ficará definida para fornecer a saída $\langle \mathcal{B} \rangle$ a partir de qualquer entrada, a parte $\mathcal{B}$ fará um trabalho único para cada entrada que receber, a fim de que, na máquina de nosso interesse, ela receba como entrada a descrição de si própria, $\langle \mathcal{B} \rangle$.

Usando então o lema relativo à máquina $\mathcal{P}_w$, descrevemos $\mathcal{B}$ a seguir.

$\mathcal{B} =$ **“** Sobre a entrada $\langle \mathcal{M} \rangle$, onde $\mathcal{M}$, uma máquina de Turing, será usada como uma porção de uma máquina de Turing:

1. Obtenha a cadeia $q(\langle \mathcal{M} \rangle) = \mathcal{P}_{\langle \mathcal{M} \rangle}$.

2. Combine a descrição $q(\langle \mathcal{M} \rangle)$ com a descrição $\langle \mathcal{M} \rangle$ de modo a formar uma máquina de Turing completa.

3. Imprima a descrição dessa nova máquina e então aceite. ”

Já vimos, pela observação 10 (seção 7.2), que podemos combinar máquinas dentro de outra máquina de Turing. Podemos afirmar, então, que está montada a parte $\mathcal{B}$. E, com ela, já podemos definir a parte $\mathcal{A}$, abaixo formalmente descrita.

$$\mathcal{A} = \mathcal{P}_{\langle \mathcal{B} \rangle}.$$

Tendo agora as duas partes, fixaremos a definição de uma máquina de Turing específica.

Definição 21. Denominamos como **AUTO** a máquina de Turing que combina as partes $\mathcal{A}$ e $\mathcal{B}$ recém-descritas. E dizemos, então, que $\langle \text{AUTO} \rangle = \langle \mathcal{A}\,\mathcal{B} \rangle$. As duas partes são combinadas do mesmo modo como $q(\langle \mathcal{M} \rangle)$ e $\langle M \rangle$ o são quando explicamos o funcionamento da máquina $\mathcal{B}$.

Pois bem, vejamos como é o funcionamento desta máquina AUTO.

1. Primeiro, $\mathcal{A}$ trabalha. Ela imprime $\langle \mathcal{B} \rangle$ na fita e volta para o começo.
2. Agora $\mathcal{B}$ trabalha. Ela recebe a entrada que é $\langle \mathcal{B} \rangle$.
3. $\mathcal{B}$ obtém $q(\langle \mathcal{B} \rangle) = \langle \mathcal{A} \rangle$, e a combina com $\langle \mathcal{B} \rangle$ de modo a formar uma nova máquina de Turing completa.
4. $\mathcal{B}$ imprime a descrição $\langle \mathcal{A}\,\mathcal{B} \rangle$ e então aceita.

Por que dissemos que $q(\langle \mathcal{B} \rangle) = \langle A \rangle$? É porque $q(\langle \mathcal{B} \rangle) = \mathcal{P}_{\langle \mathcal{B} \rangle}$, e havíamos definido a própria $\mathcal{A}$ como $\mathcal{P}_{\langle \mathcal{B} \rangle}$.

A chave da questão estava na complicação de ter um comando que fizesse referência apenas a si mesmo. Então o que montamos foi uma estrutura como a seguinte:

Imprima duas cópias do texto seguinte, sendo a segunda entre aspas.
“Imprima duas cópias do texto seguinte, sendo a segunda entre aspas.”

O comando primordial, a porção sem aspas, corresponde à parte $\mathcal{B}$ da máquina AUTO. Em teoria, poderia ter vindo qualquer conteúdo naquelas aspas abaixo, e teríamos feito o trabalho com ele. E o fato do conteúdo ser a sentença do próprio comando é o que causa a reprodução de si mesmo.

A porção de baixo, aquela com aspas, corresponde à parte $\mathcal{A}$ da máquina, que traz para ser trabalhada a partir do comando primordial a sentença dele mesmo. Ela, ao fornecer a descrição deste comando garante que o resultado seja a reprodução da própria totalidade.

Eis que, conhecendo a máquina AUTO, podemos implementar seu funcionamento em outras máquinas que precisarmos, de modo a fazê-las imprimirem suas próprias descrições e trabalharem com elas. E faremos isso com base no resultado central deste capítulo.

Teorema 11.0.2 (**Teorema da recursão**)**.** *Sejam Σ um alfabeto com que codificaremos máquinas de Turing e $t : \Sigma^* \times \Sigma^* \to \Sigma^*$ uma função computada por uma máquina de Turing $\mathcal{T}$. Então existe a uma função $r : \Sigma^* \to \Sigma^*$ computada por uma máquina $\mathcal{R}$, onde, para toda entrada $w \in \Sigma^*$, ocorre que*

$$r(w) = t\left(\langle \mathcal{R} \rangle, w\right).$$

Demonstração. Construiremos a máquina $\mathcal{R}$ de modo similar à construção da máquina AUTO. Ela terá três partes, $\mathcal{A}$, $\mathcal{B}$ e $\mathcal{T}$, sendo $\mathcal{T}$ a mesma do enunciado.

Tomemos $z \in \Sigma^*$. Havíamos definido, neste capítulo, a máquina $\mathcal{P}_z$. Sobre qualquer entrada u, ela imprimia a saída z, voltava para o começo da fita e parava. Seja $\gamma \notin \Sigma$ um símbolo. Inspirados na máquina $\mathcal{P}_z$, montamos a máquina $\mathcal{P}'_z$, cujo alfabeto de entrada é Σ. Para uma entrada $u \in \Sigma^*$ e utilizando do γ, a máquina imprime a saída $\langle u, z \rangle \in (\Sigma \cup \{\gamma\})^*$, volta para o começo da fita e para.

Vemos que tal máquina pode ser considerada associada a uma função, a qual chamamos de p'_z, de Σ^* em $(\Sigma \cup \{\gamma\})^*$. E assim como, neste capítulo, pudemos definir a função q, podemos definir uma função computável $q' : \Sigma^* \to \Sigma^*$ para a qual $q'(u) = \langle \mathcal{P}'_u \rangle$. A esta função está associada uma máquina de Turing $\mathcal{Q}'$.

Pois que, com isso, definiremos a parte $\mathcal{A}$ da máquina $\mathcal{R}$ como $\mathcal{A} = \mathcal{P}'_{\langle \mathcal{B}\mathcal{T} \rangle}$. Isto quer dizer que, sobre uma entrada $w \in \Sigma^*$, a parte $\mathcal{A}$ fornece o que seria a saída $\langle w, \mathcal{B}\mathcal{T} \rangle$, esta a partir do alfabeto $\Sigma \cup \{\gamma\}$.

E para podermos ter $\mathcal{A}$ assim, precisamos ter definido $\mathcal{B}$ e $\mathcal{T}$. Esta última já temos. Definimos a parte $\mathcal{B}$, então, como descrita a seguir. Essas duas partes são combinadas tal como explicamos a respeito da combinação das partes da máquina AUTO anteriormente neste capítulo.

$\mathcal{B} =$ “ Sobre a entrada v:

1. Examine se $v = \langle w, \mathcal{M} \rangle$, onde $w \in \Sigma^*$ e $\mathcal{M}$ é uma máquina de Turing. Se v não é desta forma, então descarte. Se é desta forma, vá para o próximo passo.

2. Distinga $\langle \mathcal{M} \rangle$ do restante da entrada e forneça a cadeia $q'(\langle \mathcal{M} \rangle)$.

3. Combine a descrição $q'(\langle\mathcal{M}\rangle)$ com a descrição $\langle\mathcal{M}\rangle$ de modo a formar uma máquina de Turing completa.

4. Forneça a saída $\langle q'(\mathcal{M})\,\mathcal{M}, w\rangle$, volte para o primeiro quadrado da fita e então aceite. ”

Pois bem. Na atividade da máquina formada respectivamente por $\mathcal{A}$, $\mathcal{B}$ e $\mathcal{T}$, primeiro acontece que, sobre uma entrada w, a parte $\mathcal{A}$ fornece a cadeia $\langle w, \mathcal{B}\,\mathcal{T}\rangle$ no início da fita e volta para o primeiro quadrado.

Depois $\mathcal{B}$ inicialmente toma apenas $\langle\mathcal{B}\,\mathcal{T}\rangle$, fornece $q'(\langle\mathcal{B}\,\mathcal{T}\rangle) = \left\langle \mathcal{P}'_{\langle\mathcal{B}\,\mathcal{T}\rangle} \right\rangle = \langle\mathcal{A}\rangle$. Essa descrição será então combinada com a própria $\langle\mathcal{B}\,\mathcal{T}\rangle$. E daí $\mathcal{B}$ fornece a cadeia $\langle\mathcal{A}\,\mathcal{B}\,\mathcal{T}, w\rangle$ e volta para o primeiro quadrado.

E, por fim, nossa máquina trabalhará como faria a máquina $\mathcal{T}$ sobre a entrada $\langle\mathcal{A}\,\mathcal{B}\,\mathcal{T}, w\rangle$ e fornecer uma saída em Σ^*. Seja, portanto, $r : \Sigma^* \to \Sigma^*$ a função cujas imagens sobre um determinado elemento são as saídas da nossa máquina de três partes sobre o mesmo elemento. Considerando $\langle\mathcal{A}\,\mathcal{B}\,\mathcal{T}\rangle = \langle\mathcal{R}\rangle$, temos que a saída $r(w)$ é igual à saída que $\mathcal{T}$ teria sobre a entrada $\langle\mathcal{R}, w\rangle$. Isto é, $r(w) = t(\langle\mathcal{R}\rangle, w)$.

Assim, r é função computável e a máquina $\mathcal{R}$ satisfaz o que procurávamos. ■

O que, no fundo, este teorema está afirmando? Que é possível termos uma máquina que obtém sua própria descrição e que pode se utilizar dela para seguir computando depois. E, a partir do momento que está demonstrado, podemos utilizar este resultado para implementar, em máquinas de Turing, a subrotina de ela obter a descrição de si mesma, e depois seguir fazendo seu trabalho.

É interessante comentar que os chamados vírus de computador, para conseguirem se espalhar entre computadores, podem usar da construção descrita na prova do teorema acima. Mas como assim? É objetivo de um vírus de computador, ao invadir um computador hospedeiro, ficar transmitindo cópias de si mesmo, isto é, de sua própria descrição algorítmica, para outras máquinas acessíveis. E conseguem fazer tal trabalho e outros através do mecanismo que acabamos de apresentamos.

Apresentaremos, por fim, dois resultados, onde aplicamos o teorema da recursão. O primeiro deles é simplesmente uma maneira mais fácil de demonstrar a indecidibilidade da linguagem do *accepting problem*.

Teorema 11.0.3. *A linguagem A_{MT} é indecidível.*

Demonstração. Suponhamos que exista uma máquina de Turing $\mathcal{H}$ que decida $\mathsf{A_{MT}}$. Ou seja, trata-se de uma máquina que somente aceita entradas $x = \langle\mathcal{M}, w\rangle$ onde a máquina de Turing $\mathcal{M}$ aceita sua entrada w. Chegaremos, em breve, a uma contradição.

Construiremos a máquina $\mathcal{B}$ a seguir.

$\mathcal{B} =$ " Sobre a entrada w:

1. Forneça, através do teorema da recursão, a descrição de si mesma, isto é, $\langle \mathcal{B} \rangle$.

2. Emule a máquina $\mathcal{H}$ sobre a entrada $\langle \mathcal{B}, w \rangle$.

3. Finalize fazendo o oposto do que $\mathcal{H}$ faria. Assim, se $\mathcal{H}$ aceita $\langle \mathcal{B}, w \rangle$, então rejeite. E se $\mathcal{H}$ rejeita $\langle \mathcal{B}, w \rangle$, então aceite. "

Ora, ou a máquina $\mathcal{B}$ aceita w, ou ela não aceita w. Deste modo, devido à subrotina $\mathcal{H}$, deveríamos sempre acabar ou em aceitação, ou em rejeição. Nunca em loop.

Mas se $\mathcal{B}$ aceita w, isto significa que $\langle \mathcal{B}, w \rangle \in \mathsf{A_{MT}}$. Assim, $\mathcal{H}$ aceita $\langle \mathcal{B}, w \rangle$, e isto implica que $\mathcal{B}$ rejeita a entrada w. Mas isto é uma contradição.

Do mesmo modo, se $\mathcal{B}$ rejeita w, isto significa que $\langle \mathcal{B}, w \rangle \notin \mathsf{A_{MT}}$. Assim, $\mathcal{H}$ rejeita $\langle \mathcal{B}, w \rangle$, e isto implica que $\mathcal{B}$ aceita a entrada w. Novamente, uma contradição.

Portanto, a existência da máquina $\mathcal{B}$ é um absurdo. E segue que a existência da máquina $\mathcal{H}$ também o é. Logo, $\mathsf{A_{MT}}$ é indecidível. ■

Percebamos a sagacidade desta demonstração perante a primeira demonstração que fizemos deste teorema.

Originalmente, havíamos construído uma máquina de Turing que, diante de cada entrada $\langle \mathcal{M} \rangle$, deveria emular a suposta $\mathcal{H}$ sobre a entrada $\langle \mathcal{M}, \langle \mathcal{M} \rangle \rangle$, e o estado de parada seria também o oposto. Entretanto, nós só notamos a contradição existente na existência desta máquina ao emularmos, sobre ela, a descrição dela mesma.

Já nesta última demonstração, na máquina que construímos, nós já temos imediatamente acesso à descrição dela mesma, seja qual for a entrada que receba. Não precisamos de uma entrada específica para perceber onde está a contradição. E isto, graças ao teorema da recursão.

E, antes do último resultado, vamos estabelecer ainda um conceito. Diremos que uma função computável é uma **transformação de descrição de máquinas de Turing** quando a imagem de uma descrição $\langle \mathcal{M}_1 \rangle$ de máquina de Turing é também a descrição $\langle \mathcal{M}_2 \rangle$ de uma máquina de Turing, e cujos demais elementos do domínio acabam sendo descartados.

O último resultado é também chamado de a versão do ponto fixo do teorema da recursão. Lembrando que um ponto fixo de uma função é um valor que não é modificado através da aplicação dela sobre si. Aqui, este "ponto fixo" está mais relacionado a manter a linguagem de uma máquina de Turing através de uma função.

Teorema 11.0.4. *Seja $t : \Sigma^* \to \Sigma^*$ uma transformação de descrição de máquinas de Turing qualquer. Então existe uma máquina de Turing $\mathcal{F}$ tal que $t(\langle\mathcal{F}\rangle)$ descreve uma máquina de Turing equivalente a $\mathcal{F}$.*

Demonstração. Sabendo que t é uma função computável, construamos a máquina $\mathcal{F}$ como abaixo.

$\mathcal{F}$ = " Sobre a entrada w:

1. Forneça, através do teorema da recursão, a descrição de si mesma, isto é, $\langle\mathcal{F}\rangle$.
2. Compute $t(\langle\mathcal{F}\rangle)$, obtendo a descrição $\langle\mathcal{G}\rangle$ de uma máquina de Turing.
3. Emule, a partir daí, a máquina $\mathcal{G}$ sobre a entrada w. "

Ora, sobre qualquer entrada w, o que a máquina $\mathcal{F}$ vai fazer, a partir de um dado momento, é emular a computação de $\mathcal{G}$ sobre w. E a máquina $\mathcal{G}$, sobre qualquer entrada w, por definição, obviamente, vai computar sobre tal entrada w. Desta maneira, as máquinas $\mathcal{F}$ e $\mathcal{G}$ terão a mesma linguagem.

Mas $\mathcal{G} = t(\langle\mathcal{F}\rangle)$. Sendo assim, as máquinas $\mathcal{F}$ e $t(\langle\mathcal{F}\rangle)$ são equivalentes. ■

12 CONSIDERAÇÕES FINAIS

Nosso trabalho se deu muito mais dentro do campo conceitual do que do campo prático. Como no exemplo do teorema a respeito da equivalência entre uma máquina de Turing multifita com alguma máquina de Turing de uma única fita. São sim vários passos a mais que a segunda usa para replicar a primeira, porém, ainda são finitos. Mas nossa concentração, aqui, se encontra justamente no ponto de, num sentido mais amplo, podermos resolver tais problemas, sem nos importarmos com a diferença de tempo.

Deparando-nos com o conceito da redutibilidade, diante de problemas indecidíveis, podemos chegar ao conhecimento de que outros problemas também são indecidíveis. Fizemos muito isso em torno do *accepting problem*. A partir de problemas dos quais sabemos que não há uma forma universal de resolver, chegamos a outros com a mesma caraterística.

E valendo-se dessa relação entre linguagens, há muito o que explorar a respeito da resolubilidade de certos problemas. Com máquinas de Turing, somos capazes de investigar certas questões nestes campos. Um exemplo disso é no âmbito do chamado ***word problem***. Para compreender, vejamos alguns conceitos.

Se um conjunto S está contido em um grupo G, então os elementos de S são **geradores** de G se, e somente se cada elemento de G pode ser escrito como um produto de finitos elementos de S e seus inversos, de modo que estão operando entre si segundo a operação definida no grupo. E dizemos que um grupo G é **finitamente gerado** por um conjunto S se, cumprindo as condições anteriores, S ainda é finito.

O produto de dois elementos $a, b \in G$ é denotado, geralmente, por ab. Assim, podemos indicar os elementos do grupo por cadeias de seus elementos. A partir do subconjunto S, formamos o grupo livre F_S, que consiste em tomar todas as cadeias formadas a partir dos elementos de S e seus inversos. Uma cadeia é também chamada de **palavra** (sobre o alfabeto $S \cup S^{-1}$). Por isso o nome de *word problem*.

Quando um elemento x está, na palavra, adjacente ao seu inverso x^{-1}, a palavra é equivalente àquela onde ambos estes elementos são omitidos. Assim, para $S = \{a, b, c\}$, as palavras $b^{-1}b^{-1}cbacc^{-1}a$ e $b^{-1}b^{-1}cbaa$ em F_S são a mesma.

Fora este tipo de redução, cada palavra de F_S é um elemento único e irrepetível por outras palavras do mesmo conjunto. Inclui-se também, no grupo livre, a cadeia vazia, que é o elemento neutro do grupo. A operação aqui é a concatenação de palavras.

Agora, tomando um conjunto de palavras de F_S, formamos um conjunto R. Seja N o menor subgrupo normal em F_S ao qual pertencem todos os elementos de R. É denominada a **apresentação** $\langle S \mid R \rangle$ o grupo quociente F_S/N. Os elementos de R são

chamados de **relações** desta apresentação.

E dizemos que o grupo G tem a apresentação $\langle S \mid R \rangle$ quando G é isomorfo ao conjunto quociente respectivo. É fácil ver que todo grupo finitamente gerado admite uma **apresentação finita**, isto é, com S e R finitos.

Sendo 1 o elemento neutro de G, podemos entender que os elementos r do conjunto N, em particular as relações em R, são aqueles elementos de F_S que são identificados com 1 no quociente. Deste modo, olhando para F_S/N, pertencem a uma mesma classe de equivalência não somente as palavras com e sem xx^{-1} em sua constituição, mas também as palavras com e sem r em sua constituição.

Tomando o mesmo conjunto S anteriormente usado, e sendo por exemplo $R = \{bb, ca\}$, então as palavras $ba^{-1}cab$ e $ba^{-1}b$ fazem parte da mesma classe de equivalência na apresentação. E, portanto, representam o mesmo elemento de G.

Ocorre que o *word problem* consiste em encontrar uma solução algorítmica que descobra se, para uma determinada apresentação, com geradores e relações finitas, duas palavras quaisquer pertencem ou não à mesma classe de equivalência.

Equivalentemente, podemos considerar o problema de decidir se dada uma palavra arbitrária sobre o alfabeto $S \cup S^{-1}$ (elemento de F_S) é equivalente ao elemento neutro de G (no quociente $F_S/N \simeq G$).

O papel das máquinas de Turing seria comparar as duas palavras. Ver em que condições especiais tal mecanismo vai conseguir decidir a respeito de toda uma classe de palavras, ou quando não é possível chegar a uma conclusão.

Ainda, tanto as máquinas de Turing como o recurso de codificar instâncias de uma forma binária são muito vistos e utilizados em teoria da complexidade computacional. Aqui apresentamos as máquinas de Turing determinísticas de uma ou mais fitas, e não-determinísticas. Mas ainda há outros modelos computacionais de serventia nesta área, por exemplo probabilísticos ou quânticos.

Todo este trabalho usando linguagens e procurando sistematizar algoritmos é aproveitado a nível teórico e prático. Persistem aí os conceitos de indecidibilidade e de redutibilidade. É trabalhada a otimização de certos procedimentos, e também o estudo de certos problemas importantes tanto dentro quanto fora da Matemática.

O leitor que tiver ficado familiarizado com o conteúdo desta dissertação já estará com um escopo importante para se debruçar nestas áreas. E nelas, poderá lidar com aspectos que aqui deixamos de lado, como o uso de recursos computacionais (tempo de execução e espaço de armazenagem) hábeis para o trabalho das máquinas.

No âmbito relativo ao recurso de tempo, encontra-se justamente um dos maiores problemas da Matemática no milênio: o **problema P *versus* NP**. Tomemos um conjunto Σ^* para tratar um problema. Dizemos que esse problema é decidível em tempo polinomial se há uma máquina cujo tempo que leva para completar sua com-

putação em cima do conjunto varia como uma função polinomial do comprimento das cadeias de entrada dele. E quando um problema é decidido em tempo polinomial, dizemos informalmente que ele foi **rapidamente decidido**.

A classe dos problemas para os quais existe um algoritmo de modo a providenciar rapidamente uma resposta sua é chamada de P. A classe dos problemas para os quais existe um algoritmo de modo a **verificar** rapidamente uma resposta é chamada de NP. E o problema P *versus* NP consiste em provar se as duas classes são iguais ou não.

Se for mostrada a igualdade, isto poderia repercutir no âmbito da criptografia de sistemas computacionais. Na Matemática em si, isto poderia facilitar a classificação e trabalho em cima de determinados problemas quando identificados em uma das duas classes. Afinal, para um problema que soubéssemos estar em NP, saberíamos automaticamente que ele está em P.

Além de que tais classes se relacionam diretamente com o objeto desta dissertação. Pois a classe NP, onde verificamos rapidamente as soluções, está associada ao conceito das máquinas de Turing não-determinísticas. Através de máquinas deste tipo verificamos em tempo polinomial que uma linguagem atende a algum requisito dentre diferentes possibilidades.

Enquanto que a classe P, onde encontramos rapidamente soluções, está associada ao conceito das máquinas de Turing determinísticas. Através de máquinas assim encontramos em tempo polinomial soluções para dados problemas. Sendo assim, tudo o que está em P, está em NP. Mas cabe verificar a recíproca.

E ainda, caso mostrada a igualdade entre as duas classes, haverá consequências na área de Logística, na qual podemos pensar a partir do conhecido **problema do caixeiro viajante**. Ele trata de descobrir o caminho ótimo para percorrer um número limitado de cidades sem passar pela mesma cidade duas ou mais vezes, poupando distância, tempo e custos.

Algebricamente, tal problema consiste no seguinte. Seja $C = \{c_1, c_2, \ldots, c_n\}$ um conjunto finito de cidades. A distância entre duas cidades c_i e c_j é $\rho_{i,j}$, sendo $\rho_{i,j} = \rho_{j,i}$ e $\rho_{i,i} = 0$. A partir disso formamos uma matriz simétrica $(\rho_{i,j})_n$ e um grafo ligando as cidades.

E seja

$$S'_n = \{\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\};\ \sigma \text{ é bijeção}$$

$$\text{tal que } \sigma(i) \neq i \text{ para todo } i \text{ do domínio}\}$$

um conjunto de permutações. A tarefa está em encontrar a permutação $\pi \in S'_n$ para a

qual a chamada **função objetivo**, $f : S'_n \to \mathbb{R}$, definida por

$$f(\sigma) = \left[\sum_{i=1}^{n-1} \rho_{\sigma(i),\sigma(i+1)}\right] + \rho_{\sigma(n),\sigma(1)},$$

possui em $f(\pi)$ seu valor mínimo.

O que esta função descreve é a distância total do circuito entre as cidades, dando uma volta inteira sem repetição. O conceito que motiva tal problema é relativamente simples. Podemos usá-lo para começar a compreender outras situações mais complexas na Logística.

E a partir da resolução de P *vs* NP, e depois, da resolução do problema do caixeiro viajante, as operações nesta área poderiam ficar mais práticas. Mas isto supondo também que conseguiríamos uma prova construtiva envolvendo tais instâncias.

E por isso, havemos de esclarecer algumas coisas também. O mais provavelmente considerado, atualmente, é que se, fosse finalizado o problema P *vs* NP, seria demonstrada a diferença entre as duas classes, e não a sua igualdade.

Além de que, mesmo que fosse demonstrada a sua igualdade, isso não significaria prontamente encontrar as resoluções para os problemas dessas classes. Pois, ainda que todos pudessem comprovadamente ser resolvidos em tempo polinomial, o grau desses polinômios pode ser tão grande quanto imaginarmos.

Fora que, como acabamos de comentar no problema do caixeiro viajante, talvez não obtivéssemos uma prova do tipo que consiga construir os algoritmos de que necessitamos, mas sim uma prova que apenas demonstra a existência de tais algoritmos. Para isso, novamente, é requisitada a dedicação em vista de, sob determinadas hipóteses, checar quando temos resoluções mais acessíveis dentro de uma classe delimitada.

Pois bem, concluímos esta dissertação, escrita de modo a expor ao leitor sobre a teoria das máquinas de Turing e suas consequências básicas. Com as descrições mais detalhadas ao longo das demonstrações, exploramos a maneira como atingíamos resultados já apontados nas bibliografias. Esperamos ajudar a abrir caminho para que outros estudantes e pesquisadores possam, então, se aprofundar em áreas afins e nelas contribuir.

13 REFERÊNCIAS

DAVIES, D. W. Corrections to Turings Universal Computing Machine. *In*: COPELAND, B. J. **The Essential Turing**: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life plus The Secrets of Enigma. New York: Oxford University Press, 2004. Disponível em: https://www.cse.chalmers.se/∼aikmitr/papers/Turing.pdf. Acesso em 05 jul. 2023.

DE MOL, L. Turing Machines. **The Stanford Encyclopedia of Philosophy**. Winter 2021 ed. Stanford: Metaphysics Research Lab, Stanford University. Disponível em: https://plato.stanford.edu/archives/win2021/entries/turing-machine/. Acesso em: 22 dez. 2023.

HÄMMERLIN, G; HOFFMANN, K. **Numerical Mathematics**. Tradução de Larry L. Schumaker. 1. ed. New York: Springer, 1991.

KLEENE, S. C. **Introduction to Metamathematics**. 7. ed. Groningen: Wolters-Noordhoff Publishing, 1974.

LIMA, E. L. **Análise Real volume 1**. Funções de uma variável. 12. ed. Rio de Janeiro: Coleção Matemática Universitária, IMPA, 2014.

PETZOLD, C. **The annotated Turing**: A Guide Tour through Alan Turing's Historic Paper on Computability and the Turing Machine. 1. ed. Indianapolis: John Wiley & Sons, 2008.

POST. E. L. Recursive Unsolvability of a problem of Thue. **Journal of Symbolic Logic**. v. 12, n. 1, p. 1-11, 1947. DOI:10.2307/2267170. Disponível em: https://www.wolframscience.com/prizes/tm23/images/Post2.pdf. Acesso em: 05 jan. 2024.

RADÓ, T. On Non-Computable Functions. **The Bell System Technical Journal**. v. 41, n. 3, p. 877-884, maio 1962. DOI: https://doi.org/10.1002/j.1538-7305.1962.tb00480.x. Disponível em: https://gwern.net/doc/cs/computable/1962-rado.pdf. Acesso em: 25 jul. 2023.

SHANNON, C. E. A universal Turing machine with two internal states. **Automata studies**. (AM-34), Princeton, v. 34, p. 157-166, 1956. DOI: https://doi.org/10.1515/9781400882618-007. Disponível em: https://www.wolframscience.com/prizes/tm23/images/

Shannon.pdf. Acesso em: 25 jul. 2023.

SIPSER, M. **Introdução à teoria da computação**. Tradução de Ruy José Guerra Barretto De Queiroz. São Paulo: Cengage Learning, 2005. Título original: Introduction to the Theory of Computation.

TURING. A. M. On computable numbers, with an application to the Entscheidungsproblem. **Proceedings of the London Mathematical Society**. London, v. s2-42, n. 1, p. 230-265, 1937. DOI: https://doi.org/10.1112/plms/s2-42.1.230. Disponível em: https://www.cs.virginia.edu/∼robins/Turing_Paper_1936.pdf. Acesso em: 14 set. 2022.

APÊNDICE A – ESPECIFICAÇÕES PARA A CONVERSÃO DE UMA MÁQUINA NÃO-DETERMINÍSTICA

Considerando a máquina $\mathcal{S}$ da demonstração do teorema 4.2.1, trazemos aqui as transições dos estados que constituem sua principal tarefa: simular os ramos da máquina não-determinística $\mathcal{N}$ dentro da segunda fita de $\mathcal{S}$, seguindo a numeração de sua terceira fita.

Continuaremos a adotar as notações dadas durante a demonstração do teorema. Assim, lembrando do que escrevemos antes, o conjunto $B_{i,\beta}$ trata-se do conjunto dos números de 1 a b que representam as tríades que estão presentes na imagem de $\delta(\mathfrak{q}_i, \beta)$.

E, tomássemos um $r \in B_{i,\beta_2}$, com $\beta_2 \in \Gamma$ qualquer. Temos que, quando a r-ésima tríade fosse ser simulada após o nó $(\mathfrak{q}_i, \beta_2)$, nós vamos chamá-la, então, de $\left(\mathfrak{q}_{f(r)}, \gamma_2, \mathsf{X}_r\right)$, onde $f : B \to \{0, 1, \ldots, n-1\}$ é a função que leva r até o número do estado de sua tríade. E definimos a função $g : B \to B$ tal que $g(b) = 1$ e $g(x) = x + 1$ para $x \neq b$.

Separamos as situações de nossas transições em oito casos, dentro dos quais explicaremos aquelas que julgamos serem as principais transições.

1. O caso em que $r \in B_{i,\beta}$, com $r \neq b$, e também $\mathfrak{q}_{f(r)} \neq \mathfrak{q}_{\text{aceita}}$ e $\mathfrak{q}_{f(r)} \neq \mathfrak{q}_{\text{rejeita}}$. Temos as transições e os estados a seguir.

   $\delta'\left(\mathfrak{q}^{r,0}_{i,\beta}, \beta_1, \beta_2, \beta_3\right) = \left(\mathfrak{q}^{r,1}_{i,\beta}, \beta_1, \beta_2, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{L}\right)$.

   $\delta'\left(\mathfrak{q}^{r,1}_{i,\beta}, \beta_1, \beta_2, r\right) = \left(\mathfrak{q}^{r,2}_{i,\beta}, \beta_1, \gamma_2, r, \mathsf{X}_r, \mathsf{X}_r, \mathsf{R}\right)$ para $\beta_2 \in \Gamma$.

   $\delta'\left(\mathfrak{q}^{r,1}_{i,\beta}, \beta_1, \dot{\beta}_2, r\right) = \left(\mathfrak{q}^{r,2}_{i,\beta}, \beta_1, \dot{\gamma}_2, r, \mathsf{X}_r, \mathsf{X}_r, \mathsf{R}\right)$ para $\beta_2 \in \Gamma$.

   $\delta'\left(\mathfrak{q}^{r,2}_{i,\beta}, \beta_1, \beta_2, \textvisiblespace\right) = \left(\mathfrak{q}^{r,3}_{i,\beta}, \beta_1, \beta_2, \textvisiblespace, \mathsf{R}, \mathsf{R}, \mathsf{L}\right)$.

   $\delta'\left(\mathfrak{q}^{r,2}_{i,\beta}, \beta_1, \beta_2, s\right) = \left(\mathfrak{q}^{s,0}_{f(r),\beta_2}, \beta_1, \beta_2, s, \mathsf{R}, \mathsf{R}, \mathsf{R}\right)$ para $\beta_2 \in \Gamma$ e $s \in B$.

   $\delta'\left(\mathfrak{q}^{r,2}_{i,\beta}, \beta_1, \dot{\beta}_2, s\right) = \left(\mathfrak{q}^{s,0}_{f(r),\beta_2}, \beta_1, \dot{\beta}_2, s, \mathsf{R}, \mathsf{R}, \mathsf{R}\right)$ para $\beta_2 \in \Gamma$ e $s \in B$.

   $\delta'\left(\mathfrak{q}^{r,3}_{i,\beta}, \beta_1, \beta_2, r\right) = (\mathfrak{q}'_9, \beta_1, \beta_2, g(r), \mathsf{L}, \mathsf{L}, \mathsf{L})$.

   É na transição do estado $\mathfrak{q}^{r,1}_{i,\beta}$ que exercemos o grande papel de imprimir o símbolo que seria impresso no ramo de $\mathcal{N}$, o símbolo aqui representado por γ_2 (ou $\dot{\gamma}_2$) no lugar de β_2 (ou $\dot{\beta}_2$). E também, o papel de andar, na segunda fita, na direção em que $\mathcal{N}$ andaria, a direção X_r.

   Na transição do estado $\mathfrak{q}^{r,2}_{i,\beta}$, o papel é o de sairmos dos estados que representam a tríade número r relacionada com o conjunto $\delta(\mathfrak{q}_i, \beta)$, para irmos para os estados $\mathfrak{q}^{s,0}_{f(r),\beta_2}$.

Aqui, s é o próximo símbolo da terceira fita, indicando a continuidade do ramo. O símbolo $f(r)$ é o número do estado que pertencia à tríade número r, a partir do qual a máquina $\mathcal{N}$ faz sua próxima transição. E β_2, que é o símbolo escaneado na segunda fita (ou escaneado em seu correspondente sobreponteado), é o próximo símbolo que $\mathcal{N}$ escanearia.

Mas isso é no caso em que a cadeia da terceira fita de fato não terminou. Se ela terminou, a transição do estado $\mathfrak{q}_{i,\beta}^{r,2}$ nos leva ao estado $\mathfrak{q}_{i,\beta}^{r,3}$, sem mudar os símbolos das três fitas. De volta ao mesmo símbolo r da terceira fita, a transição com este novo estado será tal que troca r por seu sucessor (representado por $g(r)$), não mexe consideravelmente nas outras duas fitas, e nos leva para o estado $\mathfrak{q}'_9$. Ou seja, chegamos ao ponto em que $\mathcal{S}$ recomeçará seu trabalho, explorando o próximo ramo.

2. O caso em que $r \in B_{i,\beta}$, com $r = b$, e também $\mathfrak{q}_{f(r)} \neq \mathfrak{q}_{\text{aceita}}$ e $\mathfrak{q}_{f(r)} \neq \mathfrak{q}_{\text{rejeita}}$.

   As transições a partir dos estados $\mathfrak{q}_{i,\beta}^{b,0}$, $\mathfrak{q}_{i,\beta}^{b,1}$ e $\mathfrak{q}_{i,\beta}^{b,2}$ têm a mesma estrutura que o caso anterior. A diferença está nas posteriores.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{b,3}, \beta_1, \beta_2, b\right) = \left(\mathfrak{q}_{i,\beta}^{b,3}, \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{L}\right)$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{b,3}, \beta_1, \beta_2, s\right) = (\mathfrak{q}'_9, \beta_1, \beta_2, g(s), \mathsf{L}, \mathsf{L}, \mathsf{L})$ para $s \in B$, $s \neq b$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{b,3}, \beta_1, \beta_2, \textvisiblespace\right) = \left(\mathfrak{q}_{i,\beta}^{b,4}, \beta_1, \beta_2, \textvisiblespace, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{b,4}, \beta_1, \beta_2, \beta_3\right) = \left(\mathfrak{q}_{i,\beta}^{b,4}, \beta_1, \beta_2, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$ para $\beta_3 \neq \textvisiblespace$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{b,4}, \beta_1, \beta_2, \textvisiblespace\right) = (\mathfrak{q}'_9, \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{R})$.

   Nas transições com a mesma estrutura que a anterior, a máquina simula $\mathcal{N}$ na segunda fita. E nas transições descritas logo acima, a máquina está verificando se o b da terceira fita está no final da cadeia ou não.

   Quando a máquina chega ao estado $\mathfrak{q}_{i,\beta}^{b,3}$, isso significa que o b em questão era o último símbolo da cadeia, tal como acontece no caso anterior. A máquina procura o último símbolo na terceira fita, anterior ao b, que seja distinto de b, enquanto vai mudando os símbolos b por que passa até lá por 1.

   Quando encontra este símbolo, s, o troca por seu sucessor (o símbolo $g(s)$), e passa para o estado $\mathfrak{q}'_9$ a fim de explorar o próximo ramo. Quando o que encontra, ao fim, é um quadrado vazio, é porque toda a cadeia da terceira fita era formada por símbolos b. Então, após trocar todos eles por 1 no caminho até ali, ela passa para o estado $\mathfrak{q}_{i,\beta}^{b,4}$.

   Neste estado ela procura o espaço vazio que sinaliza o final da terceira fita, enquanto vai passando por todos os símbolos dela, apenas os reimprimindo. Assim que encontra o quadrado vazio ao fim, imprime 1 no lugar, para que agora

ramos com maior comprimento sejam explorados. E, por isso, o próximo estado é $\mathfrak{q}'_9$.

3. O caso em que $r \in B_{i,\beta}$, com $\mathfrak{q}_{f(r)} = \mathfrak{q}_{\text{rejeita}}$ e $r \neq b$. A estrutura da transição a partir do estado $\mathfrak{q}^{r,0}_{i,\beta}$ é a mesma. Vejamos com seus sucessores.

   $\delta'\left(\mathfrak{q}^{r,1}_{i,\beta}, \beta_1, \beta_2, r\right) = \left(\mathfrak{q}^{r,2}_{i,\beta}, \beta_1, \beta_2, g(r), \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$.

   $\delta'\left(\mathfrak{q}^{r,2}_{i,\beta}, \beta_1, \beta_2, s\right) = \left(\mathfrak{q}^{r,2}_{i,\beta}, \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$ para $s \in B$.

   $\delta'\left(\mathfrak{q}^{r,2}_{i,\beta}, \beta_1, \beta_2, \text{␣}\right) = (\mathfrak{q}'_9, \beta_1, \beta_2, \text{␣}, \mathsf{L}, \mathsf{L}, \mathsf{L})$.

   Após trocar o r na cadeia por seu sucessor $g(r)$, à direita na terceira fita, a máquina vai para o estado $\mathfrak{q}^{r,2}_{i,\beta}$. Este substitui todos os símbolos posteriores da cadeia por 1, pois todos aqueles símbolos que viriam após o antigo r representariam ramos que seriam rejeitados.

   A transição de $\mathfrak{q}^{r,2}_{i,\beta}$ só vai para o estado $\mathfrak{q}'_9$ quando escaneia o símbolo ␣, que sinaliza o fim da terceira fita. E assim, pulamos alguns elementos da ordem lexicográfica e exploraremos outro ramo.

4. O caso em que $r \in B_{i,\beta}$, com $\mathfrak{q}_{f(r)} = \mathfrak{q}_{\text{rejeita}}$ e $r = b$. Para a transição a partir de $\mathfrak{q}^{b,0}_{i,\beta}$, a estrutura é a mesma que nos casos anteriores. A seguir, a transição com seus sucessores.

   $\delta'\left(\mathfrak{q}^{b,1}_{i,\beta}, \beta_1, \beta_2, b\right) = \left(\mathfrak{q}^{b,2}_{i,\beta}, \beta_1, \beta_2, \text{␣}, \mathsf{L}, \mathsf{L}, \mathsf{L}\right)$.

   $\delta'\left(\mathfrak{q}^{b,2}_{i,\beta}, \beta_1, \beta_2, b\right) = \left(\mathfrak{q}^{b,2}_{i,\beta}, \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{L}\right)$.

   $\delta'\left(\mathfrak{q}^{b,2}_{i,\beta}, \beta_1, \beta_2, s\right) = \left(\mathfrak{q}^{b,3}_{i,\beta}, \beta_1, \beta_2, g(s), \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$ para $s \in B$, $s \neq b$.

   $\delta'\left(\mathfrak{q}^{b,2}_{i,\beta}, \beta_1, \beta_2, \text{␣}\right) = \left(\mathfrak{q}^{b,5}_{i,\beta}, \beta_1, \beta_2, \text{␣}, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$.

   $\delta'\left(\mathfrak{q}^{b,3}_{i,\beta}, \beta_1, \beta_2, \beta_3\right) = \left(\mathfrak{q}^{b,3}_{i,\beta}, \beta_1, \beta_2, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$ para $\beta_3 \neq$ ␣.

   $\delta'\left(\mathfrak{q}^{b,3}_{i,\beta}, \beta_1, \beta_2, \text{␣}\right) = \left(\mathfrak{q}^{b,4}_{i,\beta}, \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$.

   $\delta'\left(\mathfrak{q}^{b,4}_{i,\beta}, \beta_1, \beta_2, \beta_3\right) = \left(\mathfrak{q}^{b,4}_{i,\beta}, \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$ para $\beta_3 \neq$ ␣.

   $\delta'\left(\mathfrak{q}^{b,4}_{i,\beta}, \beta_1, \beta_2, \text{␣}\right) = (\mathfrak{q}'_9, \beta_1, \beta_2, \text{␣}, \mathsf{L}, \mathsf{L}, \mathsf{L})$.

   $\delta'\left(\mathfrak{q}^{b,5}_{i,\beta}, \beta_1, \beta_2, \beta_3\right) = \left(\mathfrak{q}^{b,5}_{i,\beta}, \beta_1, \beta_2, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$ para $\beta_3 \neq$ ␣.

   $\delta'\left(\mathfrak{q}^{b,5}_{i,\beta}, \beta_1, \beta_2, \text{␣}\right) = \left(\mathfrak{q}^{b,6}_{i,\beta}, \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$.

   $\delta'\left(\mathfrak{q}^{b,6}_{i,\beta}, \beta_1, \beta_2, \beta_3\right) = \left(\mathfrak{q}^{b,6}_{i,\beta}, \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$ para $\beta_3 \neq$ ␣.

   $\delta'\left(\mathfrak{q}^{b,6}_{i,\beta}, \beta_1, \beta_2, \text{␣}\right) = (\mathfrak{q}'_9, \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{L})$.

   No estado $\mathfrak{q}^{b,1}_{i,\beta}$, a máquina imprime ␣ no lugar do b da cadeia para que, daqui a pouco, este quadrado vazio seja encontrado como a antiga posição onde se

encontrava o b. Como o estado da b-ésima tríade é de rejeição, dispensaremos todos os símbolos que vêm depois de nosso b, pois seriam rejeitados em seus ramos. Trocá-los-emos por símbolos 1. Mas temos de verificar se, até aquele b, a cadeia era formada somente por outros b ou não.

Isso fazemos no estado sucessor de $\mathfrak{q}_{i,\beta}^{b,1}$, o estado $\mathfrak{q}_{i,\beta}^{b,2}$. Ele terá transições, na questão de símbolo e direção na terceira fita, assim como do primeiro estado apresentado na explicação do caso 2 acima. Mas os estados sucessores serão diferentes.

Se, neste estado, a máquina escaneia b nesta fita, ela continua no estado. Se ela escaneia um símbolo distinto de b e de $\textvisiblespace$, ela vai para o estado $\mathfrak{q}_{i,\beta}^{b,3}$. É o caso em que não havia somente símbolos b antes na cadeia.

Tratando dessa situação, no estado $\mathfrak{q}_{i,\beta}^{b,3}$ a transição é tal que a máquina, na terceira fita, vai para a direita reimprimindo os símbolos de B que já estão lá, e mantendo-se no estado. Mas quando encontra o $\textvisiblespace$, que é onde estava o b inicialmente escaneado, ela imprime 1 no lugar, segue para a direita e muda para o estado $\mathfrak{q}_{i,\beta}^{b,4}$.

Neste estado a máquina segue indo para a direita, trocando todos os símbolos de B por 1 e mantendo-se no estado enquanto isso. Agora, quando escaneia o $\textvisiblespace$, chegou ao fim da fita. Mantém o $\textvisiblespace$ naquele quadrado e segue para o estado $\mathfrak{q}_9'$ a fim de explorar outro ramo.

Mas, de volta ao estado $\mathfrak{q}_{i,\beta}^{b,2}$, a máquina pode acabar escaneando o $\textvisiblespace$, sinalizando que havia apenas símbolos b antes na cadeia. Neste caso, o estado sucessor é $\mathfrak{q}_{i,\beta}^{b,5}$. Seu papel é quase o mesmo do estado $\mathfrak{q}_{i,\beta}^{b,3}$: mesmos símbolos impressos sobre os mesmos escaneamentos, mesmas direções, e mantém a máquina no mesmo estado enquanto escaneia símbolos de B.

Só que, quando nele a máquina escaneia o $\textvisiblespace$, onde estava o b inicialmente escaneado, o estado sucessor é $\mathfrak{q}_{i,\beta}^{b,6}$. O papel deste é quase o mesmo de $\mathfrak{q}_{i,\beta}^{b,4}$ em questão de escaneamento de símbolos de B, impressão, direção e sucessão de estado nessa situação.

Mas quando nele a máquina escaneia $\textvisiblespace$, em vez de manter tal símbolo, ela imprime 1 no lugar. Afinal, a parte anterior da cadeia, formada somente por símbolos b, e o fato de nos encontrarmos, em $\mathcal{N}$, em um estado de rejeição, nos diz que já podemos aumentar o comprimento da cadeia da terceira fita. A máquina vai, então, para o estado $\mathfrak{q}_9'$.

5. O caso em que, para $B_{i,\beta} \neq \varnothing$, temos $r \notin B_{i,\beta}$, com $r \neq b$. Tal como comentado na própria demonstração do teorema, a estrutura das transições dos estados de tal caso é igual à estrutura encontrada no caso 3.

6. O caso em que, para $B_{i,\beta} \neq \varnothing$, temos $r \notin B_{i,\beta}$, com $r = b$. Tal como comentado na própria demonstração do teorema, a estrutura das transições dos estados de tal caso é igual à estrutura encontrada no caso 2.

7. O caso em que $B_{i,\beta} = \varnothing$. Neste caso, basta definirmos o estado no qual $r = 1$. Isto porque começaremos a explorar tal caso na primeira tríade, e como ele é tal que nenhuma tríade realmente se encaixa, mal entramos e já saímos do caso.

   Vejamos a estrutura da transição de seus estados.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{1,0}, \beta_1, \beta_2, \beta_3\right) = \left(\mathfrak{q}_{i,\beta}^{1,1}, \beta_1, \beta_2, \dot{\sqcup}, \mathsf{L}, \mathsf{L}, \mathsf{L}\right)$ para $\beta_3 \neq \sqcup$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{1,0}, \beta_1, \beta_2, \sqcup\right) = \left(\mathfrak{q}_{i,\beta}^{1,1}, \beta_1, \beta_2, \sqcup, \mathsf{L}, \mathsf{L}, \mathsf{L}\right)$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{1,1}, \beta_1, \beta_2, b\right) = \left(\mathfrak{q}_{i,\beta}^{1,1}, \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{L}\right)$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{1,1}, \beta_1, \beta_2, s\right) = \left(\mathfrak{q}_{i,\beta}^{1,2}, \beta_1, \beta_2, g(s), \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$ para $s \in B$, $s \neq b$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{1,1}, \beta_1, \beta_2, \sqcup\right) = \left(\mathfrak{q}_{i,\beta}^{1,4}, \beta_1, \beta_2, \sqcup, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{1,2}, \beta_1, \beta_2, \beta_3\right) = \left(\mathfrak{q}_{i,\beta}^{1,2}, \beta_1, \beta_2, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$ para $\beta_3 \neq \sqcup$ e $\beta_3 \neq \dot{\sqcup}$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{1,2}, \beta_1, \beta_2, \dot{\sqcup}\right) = \left(\mathfrak{q}_{i,\beta}^{1,3}, \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{1,2}, \beta_1, \beta_2, \sqcup\right) = (\mathfrak{q}_9', \beta_1, \beta_2, \sqcup, \mathsf{L}, \mathsf{L}, \mathsf{L})$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{1,3}, \beta_1, \beta_2, \beta_3\right) = \left(\mathfrak{q}_{i,\beta}^{1,3}, \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$ para $\beta_3 \neq \sqcup$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{1,3}, \beta_1, \beta_2, \sqcup\right) = (\mathfrak{q}_9', \beta_1, \beta_2, \sqcup, \mathsf{L}, \mathsf{L}, \mathsf{L})$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{1,4}, \beta_1, \beta_2, \beta_3\right) = \left(\mathfrak{q}_{i,\beta}^{1,4}, \beta_1, \beta_2, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$ para $\beta_3 \neq \sqcup$ e $\beta_3 \neq \dot{\sqcup}$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{1,4}, \beta_1, \beta_2, \sqcup\right) = (\mathfrak{q}_9', \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{L})$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{1,4}, \beta_1, \beta_2, \dot{\sqcup}\right) = \left(\mathfrak{q}_{i,\beta}^{1,5}, \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{1,5}, \beta_1, \beta_2, \beta_3\right) = \left(\mathfrak{q}_{i,\beta}^{1,5}, \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{R}\right)$ para $\beta_3 \neq \sqcup$.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{1,5}, \beta_1, \beta_2, \sqcup\right) = (\mathfrak{q}_9', \beta_1, \beta_2, 1, \mathsf{L}, \mathsf{L}, \mathsf{L})$.

   No estado $\mathfrak{q}_{i,\beta}^{1,0}$, a máquina se encontra, na terceira fita, uma unidade à direita do quadrado escaneado que nortearia a simulação do atual nó de $\mathcal{N}$. "Nortearia" porque, na verdade, nenhum nó correspondente a este quadrado vai fazer qualquer diferença. Já que o nó anterior resultou, em $\mathcal{N}$, em uma imagem vazia, todos os nós posteriores a ele não fazem parte de ramos válidos.

   Bom, se no atual quadrado escaneado da terceira fita a máquina encontra o $\sqcup$, ela o mantém. Se encontra qualquer outro símbolo, o troca por $\dot{\sqcup}$. O primeiro sinaliza que ali é o fim da fita. O segundo sinaliza que não.

   Em ambos os casos a máquina, na terceira fita, volta para o quadrado correspondente ao nó atual, e o estado sucessor é $\mathfrak{q}_{i,\beta}^{1,1}$. Este estado tem aquele papel já visto de verificar se dali para trás a cadeia é formada somente por símbolos b ou não.

Se ela não for, o estado sucessor, quando encontrar um símbolo de B que não b, será $\mathfrak{q}_{i,\beta}^{1,2}$. Até aí a máquina ia para a esquerda, substituindo, se encontrasse, b por 1. E, neste momento, substituiu o símbolo distinto de b por seu sucessor.

Nesta situação, no estado $\mathfrak{q}_{i,\beta}^{1,2}$, a máquina anda para a direita, reimprimindo os símbolos de B dentro de cada quadrado e mantendo-se em tal estado. Mas ela vai encontrar, em algum momento, o símbolo impresso durante o estado $\mathfrak{q}_{i,\beta}^{1,0}$.

Se este símbolo é $\textvisiblespace$, ela o mantém e passa para $\mathfrak{q}_9'$. Se este símbolo é $\dot{\textvisiblespace}$, ela o troca por 1, vai para a direita e passa para o estado $\mathfrak{q}_{i,\beta}^{1,3}$. Neste estado, a máquina vai substituindo todos os símbolos da cadeia por 1 até chegar ao fim da fita. Ali ela escaneia $\textvisiblespace$, o mantém, e passa para o estado $\mathfrak{q}_9'$. É o pulo na ordem lexicográfica, como já vínhamos fazendo, sem aumentar o comprimento da cadeia.

Agora, retornemos ao estado $\mathfrak{q}_{i,\beta}^{1,1}$. Se nele a máquina verifica que a cadeia, antes, era formada somente por símbolos b, ela começa a andar para a direita, agora no estado $\mathfrak{q}_{i,\beta}^{1,4}$. Ela o faz reimprimindo os símbolos de B que encontra, mantendo-se neste estado. Isso até encontrar o símbolo impresso durante o estado $\mathfrak{q}_{i,\beta}^{1,0}$.

O procedimento, neste momento, é parecido com o visto no estado $\mathfrak{q}_{i,\beta}^{1,2}$. Só que aqui, se a máquina escaneia $\textvisiblespace$, ela o substitui por 1 antes de passar para o estado $\mathfrak{q}_9'$. Se escaneia $\dot{\textvisiblespace}$, ela ainda imprime 1, mas, indo para a direita, passa para o estado $\mathfrak{q}_{i,\beta}^{1,5}$.

Neste último, a máquina vai trocando tudo na cadeia por 1, indo para a direita e mantendo-se no estado. Contudo, quando escaneia $\textvisiblespace$, no fim da fita, também o troca por 1 antes de passar para o estado $\mathfrak{q}_9'$. Ou seja, os estados $\mathfrak{q}_{i,\beta}^{1,4}$ e $\mathfrak{q}_{i,\beta}^{1,5}$ servem para a situação em que iremos aumentar o comprimento da cadeia.

8. O caso em que $r \in B_{i,\beta}$, com $\mathfrak{q}_{f(r)} = \mathfrak{q}_{\text{aceita}}$. A seguir, a transição.

   $\delta'\left(\mathfrak{q}_{i,\beta}^{r,0}, \beta_1, \beta_2, \beta_3\right) = (\mathfrak{q}'_{\text{aceita}}, \beta_1, \beta_2, \beta_3, \mathsf{L}, \mathsf{L}, \mathsf{L})$.

   Simplesmente somos levados para o estado de aceitação da máquina $\mathcal{S}$.

APÊNDICE B – ADENDOS PARA A MÁQUINA UNIVERSAL

Como sabemos, a máquina universal idealizada por Turing é descrita, informalmente, por meio de uma série de m-funções e de estados não convencionais. Neste apêndice, apresentaremos alguns deles e apresentaremos também as alterações que fizemos na máquina para fundamentar a não-repetição da computação do símbolo de um mesmo F-quadrado.

Uma das m-funções é a $\mathfrak{con}$, devido a *configuration*. Ela é a responsável, na máquina, por marcar as configurações presentes tanto nas configurações completas como nas instruções. Originalmente, Turing a descrevera com apenas duas linhas no que se referia a $\mathfrak{con}_1(\mathfrak{C}, \alpha)$. Mas Post (1947) sugere a correção que aqui trazemos, onde adicionamos a terceira linha, em que se escaneia um símbolo diferente de A e de D. Esta foi pensada, realmente, para quando o símbolo escaneado durante esse estado é ␣.

Tabela 55 – Descrição a partir da m-função $\mathfrak{con}$ originalmente

Estado	Símbolo	Operações	Estado final
$\mathfrak{con}(\mathfrak{C}, \alpha)$	Não A	R, R	$\mathfrak{con}(\mathfrak{C}, \alpha)$
	A	L, Pα, R	$\mathfrak{con}_1(\mathfrak{C}, \alpha)$
$\mathfrak{con}_1(\mathfrak{C}, \alpha)$	A	R, Pα, R	$\mathfrak{con}_1(\mathfrak{C}, \alpha)$
	D	R, Pα, R	$\mathfrak{con}_2(\mathfrak{C}, \alpha)$
	Não A e não D	PD, R, Pα, R, R, R	$\mathfrak{C}$
$\mathfrak{con}_2(\mathfrak{C}, \alpha)$	C	R, Pα, R	$\mathfrak{con}_2(\mathfrak{C}, \alpha)$
	Não C	R, R	$\mathfrak{C}$

Fonte: Petzold (2008)

Acontece que essa m-função, na máquina, aparece para fazer marcar ou com y, ou com x, ou com ␣ as configurações. Só que, a partir dela, não é analisada a configuração de modo a verificar se nela está representado algum dos F-símbolos (0 ou 1). Assim, seja $\beta \in \{\mathrm{x}, \mathrm{y}\}$. A modificação que fazemos está na tabela 56.

Para $\mathfrak{con}(\mathfrak{C}, ␣)$, a descrição segue sendo a original. E para outros símbolos que vierem a ser argumentos dessas m-funções, a definição dela fica livre. O que modificamos ainda mantém a impressão dos marcadores x ou y. Mas, caso ao escanear toda a configuração a máquina verifique que o símbolo dela é o 0 ou o 1, então a cabeça agirá para começar a imprimir, no lugar do antigo marcador, a sua versão com o “chapéu”.

Em verdade, ela só chegará a substituir todos os antigos marcadores pelos novos quando estivermos usando a m-função $\mathfrak{fcon}$, acrescentada esta por nós e cuja ação na máquina é descrita na tabela 57.

Tanto para esse estado como para os que apresentaremos à frente, apenas

Tabela 56 – Descrição a partir da m-função $\mathfrak{con}$ por nós modificada

Estado	Símbolo	Operações	Estado final
$\mathfrak{con}(\mathfrak{C}, \beta)$	Não A	R, R	$\mathfrak{con}(\mathfrak{C}, \beta)$
	A	L, Pβ, R	$\mathfrak{con}_1(\mathfrak{C}, \beta)$
$\mathfrak{con}_1(\mathfrak{C}, \beta)$	A	R, Pβ, R	$\mathfrak{con}_1(\mathfrak{C}, \beta)$
	D	R, Pβ, R	$\mathfrak{con}_2(\mathfrak{C}, \beta)$
	Não A e não D	PD, R, Pβ, R, R, R	$\mathfrak{C}$
$\mathfrak{con}_2(\mathfrak{C}, \beta)$	C	R, Pβ, R	$\mathfrak{con}_3(\mathfrak{C}, \beta)$
	Não C	R, R	$\mathfrak{C}$
$\mathfrak{con}_3(\mathfrak{C}, \beta)$	C	R, Pβ, R	$\mathfrak{con}_4(\mathfrak{C}, \beta)$
	Não C	L, P$\widehat{\beta}$, L, L	$\mathfrak{fcon}(\mathfrak{C}, \beta)$
$\mathfrak{con}_4(\mathfrak{C}, \beta)$	C	R, Pβ, R	$\mathfrak{con}_5(\mathfrak{C}, \beta)$
	Não C	L, P$\widehat{\beta}$, L, L	$\mathfrak{fcon}(\mathfrak{C}, \beta)$
$\mathfrak{con}_5(\mathfrak{C}, \beta)$	C	R, Pβ, R	$\mathfrak{con}_5(\mathfrak{C}, \beta)$
	Não C	L	$\mathfrak{C}$

Fonte: Autor (2024)

Tabela 57 – Descrição a partir da função $\mathfrak{fcon}$

Estado	Símbolo	Operações	Estado final
$\mathfrak{fcon}(\mathfrak{C}, \beta)$	β	P$\widehat{\beta}$, L, L	$\mathfrak{fcon}(\mathfrak{C}, \beta)$
	Não β	R, R	$\mathfrak{C}$

Fonte: Autor (2024)

os estamos nomeando com a letra $\mathfrak{f}$ na frente para fazer referência ao fato de que eles aparecem no caso particular em que há um F-símbolo sendo representado na configuração ao fim da fita.

Com estes novos estados, ainda iremos, ao final de sua atividade, para o mesmo estado sucessor $\mathfrak{C}$ a que a máquina iria com a antiga transição para $\mathfrak{con}(\mathfrak{C}, \beta)$. E além disso, tanto $\mathfrak{fcon}(\mathfrak{C}, \beta)$ como $\mathfrak{con}_5(\mathfrak{C}, \beta)$ deixarão a cabeça da máquina em cima do símbolo marcador que foi utilizado. E isto será essencial para o que vem a seguir.

Na máquina universal, os únicos estados que realmente serão argumentos de $\mathfrak{con}$ são quatro: $\mathfrak{kom}$, $\mathfrak{kmp}$, $\mathfrak{sim}_2$ e $\mathfrak{l}(\mathfrak{l}(\mathfrak{mk}_5))$. Os dois primeiros provavelmente fazem referência à palavra *compare*. Os dois últimos serão vistos posteriormente. Os quatro são estados não convencionais, para facilitar a descrição da máquina.

O papel de $\mathfrak{kom}$ é fazer encontrar a instrução mais à direita que não estiver marcada com $\mathtt{z}$ em seu início e então imprimir tal marcador ali. Mantê-lo-emos intacto. O papel a partir de $\mathfrak{kmp}$ é o de comparar a configuração presente na instrução, marcada com símbolos $\mathtt{x}$, com a configuração presente no final da fita, marcada com $\mathtt{y}$. Com a sugestão de Davies (2004), a descrição a partir dele fica como na tabela 58.

Só que agora, com as nossas alterações, há a possibilidade de as configurações respectivas estarem marcadas com $\widehat{\mathtt{x}}$ e $\widehat{\mathtt{y}}$. Por isso, modificamos a transição de

Tabela 58 – Descrição da atividade da máquina universal sobre o estado $\mathfrak{kmp}$ originalmente

Estado	Símbolo	Operações	Estado final
$\mathfrak{kmp}$	Qualquer		$\mathfrak{cpe}(\mathfrak{e}(\mathfrak{e}(\mathfrak{anf}, \mathrm{x}), \mathrm{y}), \mathfrak{sim}, \mathrm{x}, \mathrm{y})$

Fonte: Petzold (2008)

tal estado, conforme a tabela 59.

Tabela 59 – Descrição da atividade da máquina universal sobre o estado $\mathfrak{kmp}$ por nós modificada

Estado		Símbolo	Operações	Estado final
$\mathfrak{kmp}$	{	$\widehat{\mathrm{x}}$		$\mathfrak{cpe}(\mathfrak{e}(\mathfrak{e}(\mathfrak{anf}, \mathrm{x}), \mathrm{y}), \mathfrak{fsim}, \widehat{\mathrm{x}}, \widehat{\mathrm{y}})$
	{	Não $\widehat{\mathrm{x}}$		$\mathfrak{cpe}(\mathfrak{e}(\mathfrak{e}(\mathfrak{anf}, \mathrm{x}), \mathrm{y}), \mathfrak{sim}, \mathrm{x}, \mathrm{y})$

Fonte: Autor (2024)

É por isso que fizemos com que os estados advindos das novas m-funções providenciassem que a máquina terminasse suas atividades com a cabeça exatamente acima do marcador. Pois, agora, é com este símbolo escaneável que a máquina entrará no estado $\mathfrak{kmp}$. E, dependendo de qual for o marcador, a máquina fará ou a comparação a partir dos E-símbolos x e y, ou a partir dos E-símbolos $\widehat{\mathrm{x}}$ e $\widehat{\mathrm{y}}$.

Se as sequências de F-símbolos marcadas "batessem" uma com a outra ou não, os marcadores seriam apagados de qualquer forma. E ainda serão. E se não batessem, após isso, a máquina passaria para o estado não convencional $\mathfrak{anf}$. Ainda passará, e não modificaremos seus efeitos imediatos.

Mas se elas batessem, então, após isso, a máquina passaria para o estado não convencional $\mathfrak{sim}$, devido à palavra *similar*. É o que ainda vai acontecer caso os marcadores das configurações forem aqueles que não possuem o "chapéu". A seguir, a descrição a partir dele.

Tabela 60 – Descrição da atividade da máquina universal sobre o estado $\mathfrak{sim}$ originalmente

Estado		Símbolo	Operações	Estado final
$\mathfrak{sim}$		Qualquer		$\mathfrak{f}'(\mathfrak{sim}_1, \mathfrak{sim}_1, \mathrm{z})$
$\mathfrak{sim}_1$		Qualquer		$\mathfrak{con}(\mathfrak{sim}_2, \sqcup)$
$\mathfrak{sim}_2$	{	A		$\mathfrak{sim}_3$
	{	Não A	L, Pu, R, R, R	$\mathfrak{sim}_2$
$\mathfrak{sim}_3$	{	Não A	L, Py	$\mathfrak{e}(\mathfrak{mk}, \mathrm{z})$
	{	A	L, Py, R, R, R	$\mathfrak{sim}_3$

Fonte: Petzold (2008)

A segunda linha do estado $\mathfrak{sim}_2$ já vem modificada segundo sugestão de Post, que deixou o primeiro L no lugar do que originalmente era um R. Até aí, nada que tenhamos acrescentado.

Contudo, se os marcadores forem aqueles com o "chapéu", então o próximo

estado será o que chamamos de $\mathfrak{fsim}$. Semelhante a $\mathfrak{sim}$, a descrição da atividade provocada por ele se encontra a seguir.

Tabela 61 – Descrição da atividade da máquina universal sobre o estado $\mathfrak{fsim}$

Estado	Símbolo	Operações	Estado final
$\mathfrak{fsim}$	Qualquer		$\mathfrak{f}'(\mathfrak{fsim}_1, \mathfrak{fsim}_1, \mathfrak{z})$
$\mathfrak{fsim}_1$	Qualquer		$\mathfrak{con}(\mathfrak{fsim}_2, \mathfrak{\sqcup})$
$\mathfrak{fsim}_2$	A		$\mathfrak{fsim}_3$
	Não A	L, Pu, R, R, R	$\mathfrak{fsim}_2$
$\mathfrak{fsim}_3$	Não A	L, Py	$\mathfrak{e}(\mathfrak{fm\mathfrak{k}}, \mathfrak{z})$
	A	L, Py, R, R, R	$\mathfrak{fsim}_3$

Fonte: Autor (2024)

Após a máquina encontrar a instrução correta, a partir do estado $\mathfrak{sim}$, o que ela faria seria marcar com símbolos u suas operações e com símbolos y seu estado sucessor. E também apagaria os símbolos z da fita. Ela ainda fará ambas essas coisas com o estado $\mathfrak{fsim}$. A diferença é que, com $\mathfrak{sim}$, após apagar os símbolos z, a máquina passa para o estado não convencional $\mathfrak{m\mathfrak{k}}$. E com $\mathfrak{fsim}$, a máquina passa para o estado não convencional que chamamos $\mathfrak{fm\mathfrak{k}}$.

O estado $\mathfrak{m\mathfrak{k}}$ recebe esse nome provavelmente devido à palavra *mark*. É a partir dele que a máquina faz as marcações na configuração completa ao final de sua fita com os símbolos v, x e w, como descrito a seguir.

Tabela 62 – Descrição da atividade da máquina universal sobre o estado $\mathfrak{m\mathfrak{k}}$

Estado	Símbolo	Operações	Estado final
$\mathfrak{m\mathfrak{k}}$	Qualquer		$\mathfrak{g}(\mathfrak{m\mathfrak{k}}_1, :)$
$\mathfrak{m\mathfrak{k}}_1$	Não A	R, R	$\mathfrak{m\mathfrak{k}}_1$
	A	L, L, L, L	$\mathfrak{m\mathfrak{k}}_2$
$\mathfrak{m\mathfrak{k}}_2$	C	R, Px, L, L, L	$\mathfrak{m\mathfrak{k}}_2$
	:		$\mathfrak{m\mathfrak{k}}_4$
	Não C e não :	R, Px, L, L, L	$\mathfrak{m\mathfrak{k}}_3$
$\mathfrak{m\mathfrak{k}}_3$	Não :	R, Pv, L, L, L	$\mathfrak{m\mathfrak{k}}_3$
	:	R, R	$\mathfrak{m\mathfrak{k}}_4$
$\mathfrak{m\mathfrak{k}}_4$	Qualquer		$\mathfrak{con}(\mathfrak{l}(\mathfrak{l}(\mathfrak{m\mathfrak{k}}_5)), \mathfrak{\sqcup})$
$\mathfrak{m\mathfrak{k}}_5$	Não $\sqcup$	R, Pw, R	$\mathfrak{m\mathfrak{k}}_5$
	$\sqcup$	P:	$\mathfrak{sh}$

Fonte: Petzold (2008)

O primeiro argumento na m-função $\mathfrak{g}$ do estado sucessor de $\mathfrak{m\mathfrak{k}}$ já vem corrigido por Petzold, pois originalmente ali não constava o argumento $\mathfrak{m\mathfrak{k}}_1$, e sim $\mathfrak{m\mathfrak{k}}$. Cabe destacar também que a terceira linha de $\mathfrak{m\mathfrak{k}}_2$ é destinada realmente a quando a máquina escanear o símbolo D.

Após marcar a configuração completa ao final da fita, a máquina mudaria para o estado não convencional $\mathfrak{sh}$, cujo nome se deve à palavra *show*. O objetivo de $\mathfrak{sh}$ é

fazer a máquina computar, ao final da fita, o F-símbolo que a instrução mandar. Se a instrução mandasse computar um E-símbolo, a máquina não o computaria.

Depois disso, a máquina passaria para o estado não convencional $\mathfrak{inst}$, cujo nome provavelmente se deve a *instruction* ou a *instigate*. Ele é o responsável por levar a próxima configuração completa para o fim da fita.

Ocorre que, na situação que estamos enfrentando, a instrução manda sim que se compute um F-símbolo, porém não o queremos computar. Para tal, definimos o estado $\mathfrak{fmk}$. Ele é semelhante ao estado $\mathfrak{mk}$, mas a grande diferença é que, ao final da atividade que ele derroga, a máquina não passará para o estado $\mathfrak{sh}$, e sim irá direto para o estado $\mathfrak{inst}$.

Está descrita, logo abaixo, a postura da máquina universal a partir do estado $\mathfrak{fmk}$ que acrescentamos. Assim sendo, com a definição atual, poderemos prosseguir com o que a máquina universal faria, e não teremos repetição do símbolo de um mesmo F-quadrado.

Tabela 63 – Descrição da atividade da máquina universal sobre o estado $\mathfrak{fmk}$

Estado	**Símbolo**	**Operações**	**Estado final**
$\mathfrak{fmk}$	Qualquer		$\mathfrak{g}(\mathfrak{fmk}_1, :)$
$\mathfrak{fmk}_1$	Não A	R, R	$\mathfrak{fmk}_1$
	A	L, L, L, L	$\mathfrak{fmk}_2$
$\mathfrak{fmk}_2$	C	R, Px, L, L, L	$\mathfrak{fmk}_2$
	:		$\mathfrak{fmk}_4$
	Não C e não :	R, Px, L, L, L	$\mathfrak{fmk}_3$
$\mathfrak{fmk}_3$	Não :	R, Pv, L, L, L	$\mathfrak{fmk}_3$
	:	R, R	$\mathfrak{fmk}_4$
$\mathfrak{fmk}_4$	Qualquer		$\mathfrak{con}(\mathfrak{l}(\mathfrak{l}(\mathfrak{fmk}_5)), \text{␣})$
$\mathfrak{fmk}_5$	Não ␣ e não $\widehat{\text{␣}}$	R, Pw, R	$\mathfrak{fmk}_5$
	␣ ou $\widehat{\text{␣}}$	P:	$\mathfrak{inst}$

Fonte: Autor (2024)

Ainda, os símbolos que seriam escaneados pela máquina, tanto durante o estado $\mathfrak{sh}$ como durante o estado $\mathfrak{inst}$, não serão um problema. Pois ambos esses estados são, na prática, indiferentes ao símbolo escaneado.

Nós não trouxemos a descrição da atividade a partir de $\mathfrak{sh}$ com seus estados auxiliares, e nem a partir de $\mathfrak{inst}$ com seus auxiliares. Mas basta vermos como tais descrições começam.

Tabela 64 – Descrição tabular de um exemplo de máquina de Turing

Estado	**Símbolo**	**Operações**	**Estado final**
$\mathfrak{sh}$	Qualquer		$\mathfrak{f}(\mathfrak{sh}_1, \mathfrak{inst}, \mathfrak{u})$
$\mathfrak{inst}_1$	Qualquer		$\mathfrak{g}(\mathfrak{l}(\mathfrak{inst}_1), \mathfrak{u})$

Fonte: Turing (1937)

Ora, a partir de $\mathfrak{sh}$, a máquina vai ter a mesma ação, seja qual for o símbolo escaneado. E a partir de $\mathfrak{inst}$ também. Com $\mathfrak{sh}$, não vai ter nenhuma operação que traga algo novo para o quadrado onde a cabeça se encontra. E com $\mathfrak{inst}$ também.

O que o estado sucessor de $\mathfrak{sh}$ providencia é que a cabeça procure onde está a primeira ocorrência do símbolo $\mathfrak{u}$ na fita. Então podemos dizer que a máquina "se desliga" da influência de onde a cabeça foi deixada quando passou para o estado $\mathfrak{sh}$.

E com $\mathfrak{inst}$ não é muito diferente. Pois a cabeça procurará onde está a última ocorrência do símbolo $\mathfrak{u}$ na fita. A máquina também, então, "se desligará" da influência do símbolo que escaneia quando passa para o estado $\mathfrak{inst}$.

Logo, as modificações e acréscimos que aqui fizemos se mostram estarem de acordo com a atividade e finalidade da máquina universal.

APÊNDICE C – DEMONSTRAÇÃO DO LEMA 7.1.1

Antes de começarmos a demonstração, imaginemos aqui a descrição padrão de uma máquina de Turing, dada anteriormente na dissertação, codificada nos símbolos D, A, C, R, L, E, B e ;, todos eles separados entre si por um símbolo $\widehat{\sqcup}$. E a entrada w será codificada em D e C, de acordo com a enumeração do alfabeto da fita de $\mathcal{M}$. Se tal alfabeto é $\{\alpha_0, \alpha_1, \ldots, \alpha_{m-1}\}$, então representamos α_0 por D, α_1 por DC, α_2 por DCC, e assim por diante.

O conjunto de estados de $\mathcal{M}$, sendo $\{\mathfrak{q}_0, \mathfrak{q}_1, \ldots, \mathfrak{q}_{n-1}\}$, fica convencionado com $\mathfrak{q}_0$ sendo o estado inicial. Temos que os estados de aceitação e de rejeição são representados, respectivamente, por DE e DEE. Os demais estados continuam representados de modo que $\mathfrak{q}_0$ fica como DA, $\mathfrak{q}_1$ fica como DAA, $\mathfrak{q}_2$ como DAAA, e assim por diante. Após a descrição das instruções de $\mathcal{M}$, temos os símbolos B indicando quantos são os símbolos do alfabeto de entrada. E separamos a codificação de $\mathcal{M}$ da codificação de w pelo caractere $\#$.

Usando exemplo semelhante ao usado quando explicamos a máquina universal, seja $\mathcal{M}$ tal que seu alfabeto da fita consista nos símbolos $\alpha_0 = \sqcup$, $\alpha_1 = 0$, $\alpha_2 = 1$ e $\alpha_3 = \text{ə}$. E digamos que seu alfabeto de entrada seja $\{0, 1\}$. Ela tem a seguinte descrição tabular.

Tabela 65 – Descrição tabular da máquina $\mathcal{M}$

Estado	Símbolo	Operações	Estado final
$\mathfrak{q}_0$	Qualquer	Pə, R	$\mathfrak{q}_1$
$\mathfrak{q}_1$	Qualquer	Pə, R	$\mathfrak{q}_2$
$\mathfrak{q}_2$	Qualquer	P1, R	$\mathfrak{q}_3$
$\mathfrak{q}_3$	Qualquer	P$_\sqcup$, R	$\mathfrak{q}_4$
$\mathfrak{q}_4$	Qualquer	P0, R	$\mathfrak{q}_5$
$\mathfrak{q}_5$	Qualquer	P$_\sqcup$, R	$\mathfrak{q}_2$

Fonte: Autor (2024)

Mas, aqui, vamos dizer que $\mathfrak{q}_4 = \mathfrak{q}_{\text{aceita}}$ e $\mathfrak{q}_5 = \mathfrak{q}_{\text{rejeita}}$. Sua descrição algébrica ordenada, adaptada para as condições atuais, consta abaixo.

$$\mathfrak{q}_0\alpha_0\alpha_3\text{R}\mathfrak{q}_1; \mathfrak{q}_0\alpha_1\alpha_3\text{R}\mathfrak{q}_1; \mathfrak{q}_0\alpha_2\alpha_3\text{R}\mathfrak{q}_1; \mathfrak{q}_0\alpha_3\alpha_3\text{R}\mathfrak{q}_1;$$

$$\vdots$$

$$\mathfrak{q}_5\alpha_0\alpha_0\text{R}\mathfrak{q}_2; \mathfrak{q}_5\alpha_1\alpha_0\text{R}\mathfrak{q}_2; \mathfrak{q}_5\alpha_2\alpha_0\text{R}\mathfrak{q}_2; \mathfrak{q}_5\alpha_3\alpha_0\text{R}\mathfrak{q}_2; 2$$

Em seguida, sua descrição padrão seria esta.

DADDCCCRDAA;DADCDCCCRDAA;

DADCCDCCCRDAA;DADCCCDCCCRDAA;

⋮

DEEDDRDAAA;DEEDCDRDAAA;

DEEDCCDRDAAA;DEEDCCCDRDAAA;BB

Se colocássemos, ao lado, a entrada 0010, esta ficaria representada como abaixo.

DCDCDCCDC

E, para a codificação na máquina, vamos usar, enfim, o seguinte.

; $\widehat{\sqcup}$ D $\widehat{\sqcup}$ A $\widehat{\sqcup}$ D $\widehat{\sqcup}$ D $\widehat{\sqcup}$ C $\widehat{\sqcup}$ C $\widehat{\sqcup}$ C $\widehat{\sqcup}$ R $\widehat{\sqcup}$ D $\widehat{\sqcup}$ A $\widehat{\sqcup}$ A $\widehat{\sqcup}$;

$\widehat{\sqcup}$ D $\widehat{\sqcup}$ A $\widehat{\sqcup}$ D $\widehat{\sqcup}$ C $\widehat{\sqcup}$ D $\widehat{\sqcup}$ C $\widehat{\sqcup}$ C $\widehat{\sqcup}$ C $\widehat{\sqcup}$ R $\widehat{\sqcup}$ D $\widehat{\sqcup}$ A $\widehat{\sqcup}$ A $\widehat{\sqcup}$;

⋮

$\widehat{\sqcup}$ D $\widehat{\sqcup}$ E $\widehat{\sqcup}$ E $\widehat{\sqcup}$ D $\widehat{\sqcup}$ C $\widehat{\sqcup}$ C $\widehat{\sqcup}$ D $\widehat{\sqcup}$ R $\widehat{\sqcup}$ D $\widehat{\sqcup}$ A $\widehat{\sqcup}$ A $\widehat{\sqcup}$ A $\widehat{\sqcup}$;

$\widehat{\sqcup}$ D $\widehat{\sqcup}$ E $\widehat{\sqcup}$ E $\widehat{\sqcup}$ D $\widehat{\sqcup}$ C $\widehat{\sqcup}$ C $\widehat{\sqcup}$ C $\widehat{\sqcup}$ D $\widehat{\sqcup}$ R $\widehat{\sqcup}$ D $\widehat{\sqcup}$ A $\widehat{\sqcup}$ A $\widehat{\sqcup}$ A $\widehat{\sqcup}$; $\widehat{\sqcup}$ B $\widehat{\sqcup}$ B $\widehat{\sqcup}$; #

$\widehat{\sqcup}$ D $\widehat{\sqcup}$ C $\widehat{\sqcup}$ D $\widehat{\sqcup}$ C $\widehat{\sqcup}$ D $\widehat{\sqcup}$ C $\widehat{\sqcup}$ C $\widehat{\sqcup}$ D $\widehat{\sqcup}$ C $\widehat{\sqcup}$;

Tomando essa forma para lidarmos, façamos a demonstração.

Demonstração. Convencionamos que, sempre, $\alpha_0 = \sqcup$. Dado isso, será $\mathcal{N}$ a nossa máquina modificadora.

$\mathcal{N}$ = " Sobre a entrada $\langle \mathcal{M}, w \rangle$, onde $\mathcal{M}$ é uma máquina de Turing e w é uma cadeia:

1. Leve a cabeça da máquina até a entrada w e verifique qual é o seu primeiro símbolo representado, marcando seu D e cada um de seus C com um $\mathbf{x}$ no lugar dos $\widehat{\sqcup}$ à direita. Vá para o final da fita e escreva um $\mathbf{v}$. Vá para o início da fita e marque o estado da instrução com $\mathbf{z}$. Este primeiro estado é onde há o primeiro D, e é representado ou por DA, ou por DE, ou por DEE. Vá para a próxima instrução e marque o seu estado com $\mathbf{y}$. Compare os símbolos marcados com $\mathbf{z}$ com os símbolos marcados com $\mathbf{y}$. Se baterem, vá para o fim da fita e escreva outro $\mathbf{v}$ duas unidades à direita do antigo $\mathbf{v}$. Depois substitua cada $\mathbf{y}$ por $\widehat{\mathbf{y}}$, e após o último $\widehat{\mathbf{y}}$, vá para a próxima instrução e marque seu estado com $\mathbf{y}$. Compare os marcados com $\mathbf{z}$ com os marcados com $\mathbf{y}$ e repita tal processo até não bater. Quando não bater, apague todos os $\mathbf{z}$, $\widehat{\mathbf{y}}$ e $\mathbf{y}$ da fita e siga para o

próximo passo.

2. Marque o primeiro $\mathbf{v}$ com $\mathbf{z}$. Escreva a primeira nova instrução após o fim da fita, com ␣ D ␣ A para o estado $\mathfrak{q}_0$, marque-a (ao lado dos símbolos não vazios, claro) com $\mathbf{u}$, imprima D à direita para o símbolo escaneado α_0, marque-o com $\mathbf{y}$, imprima outro D à direita, para o símbolo α_0 a ser impresso pela máquina modificada, e depois ␣ R para o movimento da direita. Compare os marcados com $\mathbf{x}$ com os marcados com $\mathbf{y}$. Se não baterem, após o final da fita imprima ␣ D ␣ E ␣ E, e depois um ␣ ;. Se baterem, transcreva o que está marcado com $\mathbf{u}$ para após o fim da fita, espaçados entre si por um quadrado cada, e adicione um ␣ A, e então um ␣ ;. Substitua os $\mathbf{u}$ por $\widehat{\mathbf{u}}$ e os $\mathbf{y}$ por $\widehat{\mathbf{y}}$.

3. Vá até o último $\mathbf{z}$ e, se houver um $\mathbf{v}$ à direita dele, marque este $\mathbf{v}$ com outro $\mathbf{z}$. Transcreva para após o fim da fita tudo o que está marcado com $\widehat{\mathbf{u}}$, espaçados entre si por um quadrado cada, e apague os $\widehat{\mathbf{u}}$ enquanto o faz. Marque este estado representado ao final com $\mathbf{u}$. Transcreva para após o fim da fita tudo o que está marcado com $\widehat{\mathbf{y}}$, espaçados entre si por um quadrado cada, apague os $\widehat{\mathbf{y}}$ enquanto o faz, e adicione um ␣ C à direita do fim. Marque este símbolo representado ao final com $\mathbf{y}$ e transcreva os marcados com $\mathbf{y}$ para após o fim da fita, espaçados entre si por um quadrado cada. Imprima ␣ R à direita e compare os marcados com $\mathbf{x}$ com os marcados com $\mathbf{y}$. Se forem diferentes, imprima ␣ D ␣ E ␣ E após o fim da fita, e depois um ␣ ;. Se forem iguais, transcreva os marcados com $\mathbf{u}$ para após o fim da fita, espaçados entre si por um quadrado cada, e adicione um ␣ A, e depois um ␣ ;. Substitua os $\mathbf{u}$ por $\widehat{\mathbf{u}}$ e os $\mathbf{y}$ por $\widehat{\mathbf{y}}$. E então volte para ver se há um $\mathbf{v}$ ao lado do último $\mathbf{z}$. Quando não há outro $\mathbf{v}$, apague os marcadores $\mathbf{z}$ e os marcadores $\widehat{\mathbf{y}}$, e siga para o próximo passo.

4. Substitua os $\mathbf{x}$ (que marcavam o primeiro símbolo de w) por $\widehat{\mathbf{x}}$ e, se houver, marque o próximo símbolo representado de w com $\mathbf{x}$. Procure o primeiro $\mathbf{v}$ da fita e o marque com $\widehat{\mathbf{z}}$. Então transcreva tudo o que está marcado por $\widehat{\mathbf{u}}$ para após o fim da fita, espaçados entre si por um quadrado cada, apagando os $\widehat{\mathbf{u}}$ enquanto isso, e adicionando um ␣ A após o fim da fita. Marque este estado representado ao final com $\mathbf{u}$. Imprima D à direita e o marque com $\mathbf{y}$. Transcreva tudo o que está marcado com $\mathbf{y}$ para após o fim da fita, espaçados entre si por um quadrado cada, imprima ␣ R à direita e compare os marcados com $\mathbf{x}$ com os marcados com $\mathbf{y}$. Se diferentes, escreva ␣ D ␣ E ␣ E após o fim da fita, e depois um ␣ ;. Se iguais, transcreva os marcados com $\mathbf{u}$ para após o fim da fita, espaçados entre si por um quadrado cada, e adicione um ␣ A à direita, e depois um ␣ ;. Substitua os $\mathbf{u}$ por $\widehat{\mathbf{u}}$ e os $\mathbf{y}$ por $\widehat{\mathbf{y}}$.

5. Vá até o último $\mathbf{z}$ da fita e repita o processo do item 3. Quando não se encontrar

mais novos $\mathbf{v}$, repita o processo do item 4. Quando tiver feito as instruções do estado referente ao último símbolo de w, não haverá mais símbolos representados logo depois do último marcador $\mathbf{x}$. A cabeça, indo para a direita procurando-o, mudará de comportamento quando encontrar, no lugar disso, um $\mathbf{v}$. Aí, a máquina apagará todos os marcadores $\mathbf{x}$ e $\widehat{\mathbf{x}}$. Marque com $\mathbf{x}$ o estado da última instrução no fim da fita, em seu D e seus A. Marque o primeiro $\mathbf{v}$ com $\mathbf{z}$. Transcreva para após o fim da fita os marcados com $\mathbf{x}$, espaçados entre si por um quadrado cada. Marque o estado representado ao final da fita com $\mathbf{u}$, imprima D à direita, marque-o com $\mathbf{y}$ e transcreva o marcado com $\mathbf{y}$ para após o final da fita. Imprima ␣ L à direita. Transcreva o marcado com $\mathbf{u}$ para o fim da fita e adicione um ␣ A e um ␣ ;. Substitua os $\mathbf{u}$ por $\widehat{\mathbf{u}}$ e os $\mathbf{y}$ por $\widehat{\mathbf{y}}$.

6. Vá até o último $\mathbf{z}$ e, se houver um $\mathbf{v}$ à direita dele, marque este $\mathbf{v}$ com outro $\mathbf{z}$. Transcreva para após o fim da fita tudo o que está marcado com $\widehat{\mathbf{u}}$, espaçados entre si por um quadrado cada, e apague os $\widehat{\mathbf{u}}$ enquanto o faz. Marque este estado representado ao final com $\mathbf{u}$. Transcreva para após o fim da fita tudo o que está marcado com $\widehat{\mathbf{y}}$, espaçados por um quadrado cada, apague os $\widehat{\mathbf{y}}$ enquanto o faz, e adicione um ␣ C após o fim. Marque este símbolo representado ao final com $\mathbf{y}$ e transcreva os marcados com $\mathbf{y}$ para após o fim da fita, espaçados por um quadrado cada. Imprima ␣ L após o fim, escreva ␣ D ␣ E ␣ E e depois escreva um ␣ ;. Substitua os $\mathbf{u}$ por $\widehat{\mathbf{u}}$ e os $\mathbf{y}$ por $\widehat{\mathbf{y}}$.

7. Repita o item 6 até não haver mais novos $\mathbf{v}$.

8. Quando não houver mais novos $\mathbf{v}$, apague todos os $\mathbf{z}$. Procure, comparando, qual é a última instrução antes da marcada com $\mathbf{x}$, que possui o seu estado enumerado menor do que o estado marcado com $\mathbf{x}$. Ao encontrá-lo, marque-o com $\mathbf{x}$ e apague as marcações de $\mathbf{x}$ que já existiam antes. Marque o primeiro $\mathbf{v}$ com $\mathbf{z}$ e então transcreva, para após o fim da fita, espaçados entre si por um quadrado cada, os marcados com $\widehat{\mathbf{u}}$, apagando os $\widehat{\mathbf{u}}$ enquanto isso. Adicione um ␣ A após o fim e marque esta representação de um estado com $\mathbf{u}$. Imprima D à direita e o marque com $\mathbf{y}$. Transcreva o que foi marcado com $\mathbf{y}$ para após o fim da fita, espaçados entre si por um quadrado cada. Imprima ␣ L à direita e transcreva o marcado por $\mathbf{u}$ para após o fim da fita, adicionando um ␣ A à direita, e depois um ␣ ;. E substitua os $\mathbf{u}$ por $\widehat{\mathbf{u}}$ e os $\mathbf{y}$ por $\widehat{\mathbf{y}}$.

9. Vá até o último $\mathbf{z}$ e, se houver um $\mathbf{v}$ à direita dele, marque este $\mathbf{v}$ com outro $\mathbf{z}$. Transcreva para após o fim da fita tudo o que está marcado com $\widehat{\mathbf{u}}$, espaçados entre si por um quadrado cada, e apague os $\widehat{\mathbf{u}}$ enquanto o faz. Marque este estado representado ao final com $\mathbf{u}$. Transcreva para o fim da fita tudo o que está marcado com $\widehat{\mathbf{y}}$, espaçados entre si por um quadrado cada, apague os $\widehat{\mathbf{y}}$ en-

quanto o faz, e adicione um ␣ C após o fim. Marque este símbolo representado ao final com $\mathbf{y}$ e transcreva os marcados com $\mathbf{y}$ para após o fim da fita, espaçados entre si por um quadrado cada. Imprima ␣ L à direita. Transcreva para o fim da fita o que está marcado com $\mathbf{u}$, espaçados entre si por um quadrado cada, adicione um ␣ A à direita, e depois um ␣ ;. Substitua os $\mathbf{u}$ por $\widehat{\mathbf{u}}$ e os $\mathbf{y}$ por $\widehat{\mathbf{y}}$.

10. Repita o item 9 até não haver mais novos $\mathbf{v}$. Repita o item 8 até que se marque com $\mathbf{x}$ as instruções do primeiro estado. Após o processo se repetir com este, volte para a esquerda procurando estados e encontre o símbolo $\#$, que é sinal para parar com tal procedimento. Apague todos os marcadores $\mathbf{x}$, os $\widehat{\mathbf{u}}$ e os $\widehat{\mathbf{y}}$. Vá para o fim da fita e marque os A do estado da última instrução com $\mathbf{y}$.

11. Vá para o início da fita e marque com $\mathbf{x}$ o estado da primeira instrução, seja ele representado com A ou em E (no caso do estado de aceitação ou do de rejeição), e marque com $\mathbf{z}$ todo o restante da instrução. Transcreva o que foi marcado com $\mathbf{x}$ para o final da fita, espaçados entre si por um quadrado cada, enquanto substitui cada $\mathbf{x}$ por $\widehat{\mathbf{x}}$. Verifique se o último símbolo da fita é A ou E. Se A, procure o último $\mathbf{y}$, substitua-o por $\widehat{\mathbf{y}}$ e adicione um ␣ A após o fim da fita. Procure novamente o último $\mathbf{y}$ e repita o processo até não haver mais $\mathbf{y}$ na fita. Então siga em frente com o próximo passo. Se E, substitua todos os $\mathbf{y}$ da fita por $\widehat{\mathbf{y}}$ e siga em frente com o próximo passo.

12. Transcreva tudo o que foi marcado com $\mathbf{z}$ para após o final da fita, espaçados entre si por um quadrado cada, enquanto substitui cada $\mathbf{z}$ por $\widehat{\mathbf{z}}$. Verifique se o último símbolo da fita é A ou E. Se A, procure o último $\widehat{\mathbf{y}}$, substitua-o por $\mathbf{y}$ e adicione um ␣ A após o fim da fita. Procure novamente o último $\widehat{\mathbf{y}}$ e repita o processo até não haver mais $\widehat{\mathbf{y}}$ na fita. Adicione um ␣ ; após o fim da fita e siga em frente com o próximo passo. Se E, substitua todos os $\widehat{\mathbf{y}}$ da fita por $\mathbf{y}$, adicione um ␣ ; após o fim da fita e siga em frente com o próximo passo.

13. Procure o último $\widehat{\mathbf{z}}$ da fita e marque com $\mathbf{x}$, na instrução imediatamente posterior à dele, o estado, e com $\mathbf{z}$ o restante desta instrução. Apague os $\widehat{\mathbf{x}}$ e $\widehat{\mathbf{z}}$ da fita e transcreva tudo o que foi marcado com $\mathbf{x}$ para após o final da fita, espaçados entre si por um quadrado cada. Será repetido o mesmo processo feito anteriormente a partir da transcrição dos marcados por $\mathbf{x}$ no item 11. O procedimento para quando, após encontrar o último $\widehat{\mathbf{z}}$ da fita e estiver procurando o estado da próxima instrução, a cabeça se deparar com o símbolo $\#$.

14. Marque os B da fita com $\mathbf{z}$ e depois transcreva tais símbolos marcados para após o final da fita, espaçados entre si por um quadrado cada. Adicione um ␣ ; após o fim da fita.

15. Apague tudo que há antes do # na fita. Apague tudo a partir do # até antes do primeiro D que aparecer."

Esta máquina faz a modificação necessária na descrição da máquina $\mathcal{M}$. A contagem inicial dos $\mathbf{v}$ serve para saber o tamanho do alfabeto de $\mathcal{M}$. Os primeiros estados impressos se guiam pela quantidade de $\mathbf{z}$, para que cada estado seja esquematizado segundo cada símbolo do alfabeto. O símbolo é comparado com o primeiro símbolo de w para que sempre o reconheça, e rejeite todos os outros. Isto está representado até o item 3.

Se o estado reconhece o símbolo, andamos para a direita e teremos o próximo estado, fazendo o mesmo com o próximo símbolo de w. E assim também com o próximo, até que cheguemos ao último símbolo de w e o último desses primeiros estados.

O primeiro estado logo depois disso tem uma tarefa singular. Ele deve verificar se o símbolo à direita do último símbolo de w é realmente o símbolo $\llcorner$. Porque, se não, a nossa cadeia não será realmente w. Isso é descrito nos itens 5 e 6, onde todos os símbolos recebem rejeição, exceto α_0. E aqui já começamos a andar para a esquerda.

Agora, os estados precisam fazer a cabeça da máquina retornar ao primeiro símbolo após verificar que a cadeia é a que nos interessa. Para isso, ela está retornando, com o mesmo número de passos, para o primeiro quadrado. É isso que está fazendo enquanto acompanha a regressão de marcações em $\mathbf{x}$ dos primeiros estados. Dos itens 8 até o 10 se está percorrendo tal caminho.

Esses foram os estados adicionados na descrição da máquina $\mathcal{M}$. Agora devemos escrever os estados originais dela, aqueles que já constam na descrição codificada na entrada. Então, o que fazemos é trazê-los para o final da fita. Os novos estados são aqueles com que a máquina $\mathcal{M}$ inicia trabalhando agora, e após todos eles, vem o que seria o antigo estado $\mathfrak{q}_0$ de $\mathcal{M}$.

Assim, além de simplesmente realocarmos para a posição mais à direita na fita, temos que ajustar a numeração dos estados originais. Em verdade o que vai acontecer é que todos terão seu número acrescido da mesma quantidade, que é a quantidade de novos estados. E esta quantidade é também o número do último estado computado pela máquina no fim da fita.

E o que resolvemos fazer, mostrado no final do item 10 até o 13, foi marcar tal número de símbolos A com a letra $\mathbf{y}$, para saber quantos A adicionar enquanto transcrevíamos cada instrução original para o fim da fita. Fazemo-lo tanto no estado da instrução quanto no estado sucessor, tomando o cuidado para não o fazer nos estados de aceitação e de rejeição, que não têm marcação com A.

O item 14 transcreve os símbolos B para o fim da nova descrição, de modo a indicar os símbolos que fazem parte do alfabeto de entrada, que ainda são os mesmos.

Por fim, apagamos tudo o que há antes das novas instruções, sobrando apenas elas mesmas. E estas instruções já descrevem a máquina que rejeita todas as cadeias que não w, e que age como $\mathcal{M}$ agiria diante de tal cadeia. ■