

# **Controle de Tráfego Urbano Inteligente**

Trabalho Final do Curso de Teoria da Computação

João Roberto  
Caio Mello  
Jonas Oliveira  
Leandro Barbosa  
Vinicius Caires

**Período Letivo**  
2025.2

# Índice

1. Introdução .....	3
1.1. Enunciado do problema .....	3
1.2. Contextualização e importância na área de cidades inteligentes. ....	3
1.3. Abstrações, notações específicas e corolários significativos .....	4
2. Máquina de uma única fita ( $MT_1$ ) .....	6
2.1. O que são os módulos de uma MT? .....	6
2.2. Módulos em $MT_1$ .....	7
2.2.1. Submódulo $\psi$ (ou “função de contagem”) .....	8
2.2.2. Módulo $\alpha$ .....	11
2.2.3. Do custo da contagem no módulo $\alpha$ .....	13
2.2.3.1. Análise para $w = y^k * x^m$ .....	14
2.2.3.2. Análise para $w = x^m y^k$ .....	14
2.2.4. O custo da limpeza .....	16
2.2.5. Conclusão do custo de $\alpha$ .....	16
2.2.6. Módulo $\alpha$ na íntegra .....	17
2.2.7. Módulo $\beta$ .....	18
2.2.8. Função $\psi'$ .....	19
2.2.8.1. Prova da certeza de $\psi'$ .....	19
2.2.8.2. Análise temporal de $\psi'$ .....	21
2.2.8.3. Custo total da $MT_1$ .....	22
2.2.9. Módulo $\beta$ na íntegra .....	23
2.3. Máquina na íntegra .....	24
2.4. Exemplos passo a passo .....	25
2.4.1. Exemplo 1 .....	25
2.4.2. Exemplo 2 .....	25
3. Máquina de múltiplas fitas ( $MT_2$ ) .....	25
3.1. Definição formal e organização em módulos .....	26
3.2. Representação das fitas .....	27
3.3. Funcionamento da máquina .....	27
3.4. Análise formal de complexidade .....	28
3.4.1. Leitura da entrada e contagem .....	28
3.4.2. Construção da fita do total de veículos .....	29
3.4.3. Cálculo da metade do total .....	29
3.4.4. Verificação de maioria absoluta e comparações .....	29
3.4.5. Escrita da saída e encerramento .....	30
3.4.6. Custo total no pior caso .....	30
3.4.7. Complexidade em número de estados .....	31
3.5. Exemplos passo a passo .....	31
3.5.1. Exemplo 1 .....	31
3.5.2. Exemplo 2 .....	32
3.6. Comparação com a $MT_1$ .....	33
4. Considerações finais .....	34

# 1. Introdução

## 1.1. Enunciado do problema

*A Companhia Turing Traffic Systems está desenvolvendo um sistema de controle de tráfego urbano baseado em Máquinas de Turing para gerenciar os semáforos de uma cidade inteligente. O objetivo é determinar qual deve ser o próximo estado do semáforo principal de um cruzamento, considerando informações sobre o fluxo de veículos.*

*O cruzamento possui três avenidas que se encontram no mesmo ponto, e cada avenida possui sensores que contam o número de veículos aguardando em cada direção.*

*O sistema deve decidir a próxima cor do semáforo principal de acordo com a seguinte política:*

***I** - Se uma avenida tiver mais da metade do total de veículos somados das três avenidas, ela terá prioridade, e o semáforo para essa avenida ficará verde, enquanto os outros ficarão vermelho.*

***II** Caso nenhuma avenida tenha mais da metade dos veículos, o sistema escolhe a avenida com maior número de veículos.*

***III** Em caso de empate, a avenida A tem prioridade sobre B, e B sobre C (ordem  $A > B > C$ ).*

## 1.2. Contextualização e importância na área de cidades inteligentes.

O crescimento acelerado dos centros urbanos e o aumento do fluxo de veículos tornam indispensável o uso de sistemas inteligentes para o gerenciamento do tráfego urbano. Sistemas tradicionais de semáforos, baseados em intervalos fixos de tempo, não conseguem se adaptar dinamicamente às variações do fluxo de veículos, impactando negativamente a qualidade de vida da população, aumentando o consumo de combustível, elevando os níveis de poluição atmosférica e comprometendo a eficiência dos sistemas de transporte público.

As cidades inteligentes (smart cities) propõem o uso integrado de tecnologias da informação, comunicação e automação para otimizar serviços urbanos e promover sustentabilidade, segurança e eficiência operacional. Sistemas inteligentes de controle de tráfego desempenham um papel fundamental, pois permitem a adaptação em tempo real dos semáforos conforme a demanda de veículos, reduzindo tempos de espera e melhorando a fluidez do trânsito nos cruzamentos.

O presente trabalho aborda o desenvolvimento de um sistema de controle de tráfego urbano inteligente baseado em Máquinas de Turing, um modelo formal clássico da computação. Embora abstratas, as Máquinas de Turing permitem a especificação rigorosa de algoritmos e processos de decisão, garantindo correção lógica, previsibilidade e análise formal de complexidade. O objetivo é decidir qual avenida de um cruzamento deve receber o sinal verde, com base no número de veículos aguardando em cada via.

Ao considerar um cruzamento com três avenidas monitoradas por sensores de contagem de veículos, o sistema proposto demonstra como regras de prioridade podem ser formalizadas computacionalmente para determinar, de forma eficiente, qual via deve receber o sinal verde. Essa abordagem leva em conta princípios fundamentais das cidades inteligentes, como o uso de dados em tempo real, a automação de decisões e a busca pela otimização dos recursos urbanos.

Dessa forma, o trabalho contribui tanto para a compreensão prática da aplicação de Máquinas de Turing em problemas do mundo real quanto para a discussão de soluções computacionais voltadas à mobilidade urbana inteligente, evidenciando a relevância da computação teórica como base para o desenvolvimento de sistemas urbanos modernos e eficientes.

### 1.3. Abstrações, notações específicas e corolários significativos

Como parte desse trabalho, temos a missão de modular Máquinas de Turing que solucionem o problema preposto. Para tal, teremos como entrada uma string representando a distribuição de carros dentre as avenidas A, B e C, seguindo o formato:

Seja  $w$  uma palavra e  $n \in \mathbb{N}$

$$w = \varphi_0 * \dots * \varphi_{n-1} * \varphi_n$$

$$\varphi_k \in \{a, b, c\} \text{ tal que } 0 \leq k \leq |w|$$

Nesta representação, cada carro na avenida A será uma letra ‘a’, na avenida B, a letra ‘b’, e na C, ‘c’.

Uma outra definição útil é a do conjunto de ocorrências de um conjunto de símbolos ou de um único símbolo (sendo esse o caso do conjunto unário).

Seja  $w$  uma palavra,  $x$  uma letra e  $X = \{x\}$

$X_w$  é o conjunto de ocorrências dos símbolos de  $X$  em  $w$

$$X_w = \{j \mid \forall j \in \mathbb{N} \text{ tal que } 0 \leq j \leq |w| \text{ e } \varphi_j \in X\}$$

Será útil também a definição de uma função para cálculo da cardinalidade de uma letra, descrito como segue:

Seja  $w$  uma palavra e  $x$  uma letra

$$\gamma_x(w) : \{a, b, c\} \rightarrow \mathbb{N}$$

$$\gamma_x(w) = |X_w|$$

De forma direta,  $\gamma_x(w)$  nos retorna o número de ocorrências da letra  $x$  na palavra  $w$ .

Retornando ao enunciado da questão, vamos destrinchar os comandos escritos e abstrair duas propriedades relevantes.

1. Determinar a avenida com quantidade de carros superior a metade do total de automóveis em todas avenidas.

1.1 Ou seja, **determinar a letra  $x$  cuja cardinalidade é superior ao tamanho total da string  $w$  de entrada:**

$$x \in \{a, b, c\} \mid \gamma_x(w) > \frac{|w|}{2}$$

dado que as únicas letras de  $w$  são  $\{a, b, c\}$ , segue que:

$$|w| = \gamma_a(w) + \gamma_b(w) + \gamma_c(w)$$

$$\text{Portanto, } \gamma_x(w) > \frac{|w|}{2} \Leftrightarrow \gamma_x(w) > \frac{\gamma_a(w) + \gamma_b(w) + \gamma_c(w)}{2}$$

1.2 Caso mais de uma avenida se qualifique, devemos respeitar a A sobre B, B sobre C e, por transitividade, A sobre C.

2. Determinar a avenida com a maior quantidade de carros.

2.1 Ou seja, **determinar a letra  $x$  de maior cardinalidade na palavra  $w$ :**

$$x, y, z \in \{a, b, c\} \wedge y \neq z \wedge z \neq x \wedge x \neq z$$

$$\gamma_x(w) > \gamma_y(w) \wedge \gamma_x(w) > \gamma_z(w)$$

2.2 Em caso de empate, devemos respeitar a ordem de prioridade definida em 1.2.

Também é importante considerar que a computação da [propriedade 2](#) só deve ser avaliada caso não haja parada por estado final na computação da [condição 1](#).

Vamos considerar também algumas equivalências para facilitar a modelagem e validação do automôto.

A primeira é centrada na [propriedade A](#), a qual nos dá que:

$$\gamma_x(w) > \frac{\gamma_a(w) + \gamma_b(w) + \gamma_c(w)}{2}$$

$$2 * \gamma_x(w) > \gamma_a(w) + \gamma_b(w) + \gamma_c(w)$$

Dado que  $x \in \{a, b, c\}$ ,

$$\gamma_x(w) \in \{\gamma_a(w), \gamma_b(w), \gamma_c(w)\}$$

Generalizando,

$$\text{Sejam } x, y, z \in \{a, b, c\} \mid x \neq y \wedge y \neq z$$

$$2 * \gamma_x(w) > \gamma_x(w) + \gamma_y(w) + \gamma_z(w)$$

$$\gamma_x(w) > \gamma_y(w) + \gamma_z(w) \quad (\Delta)$$

Ou seja,

$$\gamma_x(w) > \frac{\gamma_x(w) + \gamma_y(w) + \gamma_z(w)}{2} \Leftrightarrow \gamma_x(w) > \gamma_y(w) + \gamma_z(w)$$

Portanto, basta que nosso automômo compute o [corolário Δ](#):

$$\gamma_x(w) > \gamma_y(w) + \gamma_z(w)$$

para que também seja verificada a [condição 1](#) do enunciado e portanto determinada a avenida prioritária.

## 2. Máquina de uma única fita (MT<sub>1</sub>)

Nossa máquina de uma fita só, ou MT<sub>1</sub>, é definida como:

$$MT_1 = (E, \Sigma, \Gamma, \delta, i, F)$$

Tal que,

$$\Sigma = \{a, b, c\}$$

$$\Gamma = \{x_{\#} \mid \forall x \in \Sigma\} \cup \{\varepsilon, <\} \cup \Sigma, \text{ ou seja, } \{a, b, c, a_{\#}, b_{\#}, c_{\#}, <, \varepsilon\}$$

$$F = \{F_a, F_b, F_c\}$$

< é o símbolo delimitador do início da fita. Tratamos a fita como infinita a direita.

ε representa o vazio que preenche os espaços à direita da palavra inserida.

Cada estado final representa a decisão de conceder prioridade a uma avenida no cruzamento, onde:

$$\text{Sendo } x \in \Sigma \mid F_x : \text{Decisão de priorizar a avenida } x$$

As transições δ, estados E, F e i serão definidos em representações gráficas mais a frente.

MT<sub>1</sub> tem dois módulos principais: α e β os quais serão estabelecidos e destrinchados em seções posteriores.

### 2.1. O que são os módulos de uma MT?

Com objetivo de tratar da complexidade do problema de maneira didática, facilitando nossa argumentação, além de simplificar o processo de desenvolvimento da máquina abstrata, vamos segmentar porções de MT's (Máquinas de Turing) em 'módulos'.

Definida uma MT como:

$$MT = (E, \Sigma, \Gamma, \delta, i, F)$$

E: Conjunto finito de estados

Σ: Alfabeto de entrada

Γ: Alfabeto da fita

δ: Funções de transição

i: Estado inicial

F: Conjunto de estados finais

O módulo  $\mathbb{M}$  de  $MT$  é:

$$\mathbb{M} = (E_\xi, \Sigma, \Gamma, \delta_\xi, e_{in}, E_{out})$$

Onde,

$E_\xi$ : Subconjunto de estados em  $E$ .

$\delta_\xi$ : Subconjunto de transições em  $\delta$  que atuem somente sobre os estados definidos em  $E_\xi$ .

$e_{in}$ : Um estado eleito de  $E_\xi$ , deve denotar o início do módulo. Vamos nos referir a ele como estado de entrada.

$e_{out}$ : Um conjunto de estados finais pertencentes a  $E_\xi$ , representam o fim da computação do módulo. Chamaremos eles de estados de saída.

Importante notar que não necessariamente  $e_{out} \subset F$ . Ou seja, o fim da computação de  $\mathbb{M}$  não é necessário e suficiente para parada da  $MT$ .

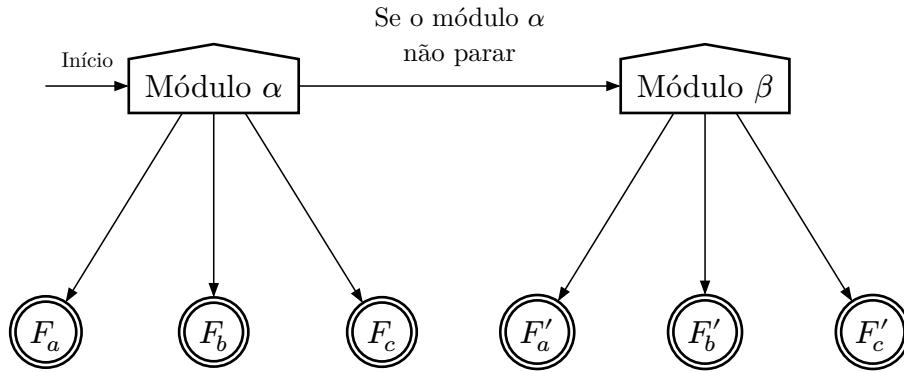
No decorrer desse texto também usaremos o termo “submódulo”. Essa estrutura é análoga à um módulo e sua definição é construída em função de um módulo  $\mathbb{M}$  ao invés da  $MT$  por inteiro. Essa outra abstração também tem finalidades puramente organizacionais.

## 2.2. Módulos em $MT_1$

O módulo  $\alpha$  será responsável por verificar se a [condição 1](#) do enunciado vale para alguma letra de  $\Sigma$  através do [corolário  \$\Delta\$](#) .

Caso nenhum estado final seja alcançado dentro de  $\alpha$ , o automôto recorrerá ao módulo  $\beta$ , onde avaliará a regra de desempate definida na [condição 2](#) do problema. Verificando qual símbolo tem a maior cardinalidade na entrada.

A macroarquitetura da máquina assume o formato:



Fluxograma 1: Arquitetura do automôto

### 2.2.1. Submódulo $\psi$ (ou “função de contagem”)

A peça central de MT1 é o submódulo  $\psi$ , o qual também chamaremos de função de contagem, responsável por comparar a cardinalidade de diferentes conjuntos de símbolos em uma dada palavra.

Definindo formalmente,

$$L \subset \Gamma$$

$L_w, X_w$  são os conjuntos de ocorrências dos elementos de  $L$  e de  $x$  em  $w$

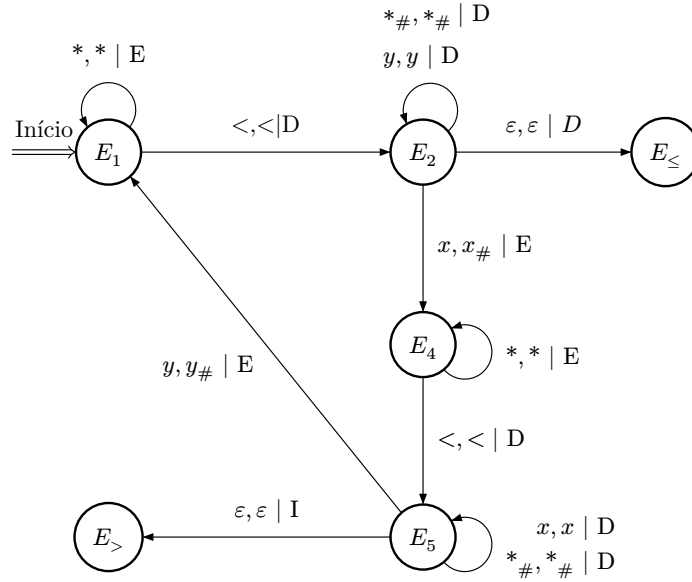
$\psi(x, L)$  : Avalia a natureza do produto cartesiano  $X_w \times L_w$

Verificando a ocorrência de bijeção, injeção e/ou sobrejeção, abstraímos sentidos de acordo ao definido pela Teoria de Conjuntos, em caso de:

1. bijeção, intuimos que  $\gamma_x(w) = \sum_{y \in L} \gamma_y(w)$ ;
2. injeção, mas não sobrejeção,  $\gamma_x(w) < \sum_{y \in L} \gamma_y(w)$ ;
3. sobrejeção, mas não injeção,  $\gamma_x(w) > \sum_{y \in L} \gamma_y(w)$

Como as regras de negócio estipuladas ao sistema de acordo ao enunciado da questão preocupam-se apenas com o que já definimos em  $\Delta$ , o submódulo  $\psi$  vai ter apenas dois estados de aceitação ou parada, um para bijeção ou sobrejeção (1 ou 3) e outra para injeção (2).

A título de demonstração, tratemos  $L = \{y\}$ ,  $\psi(x, L)$  assume o formato:



Submódulo 1: Função  $\psi$

Os dois estados de saída ou parada desse submódulo serão  $E_{<}$  e  $E_{\geq}$ , os quais semanticamente representam:

- $E_{<}$ :  $X_w < L_w$
- $E_{\geq}$ :  $X_w \geq L_w$



Naturalmente, em caso de  $|L| > 1$ , as transições que contenham símbolos de  $L$  na representação acima, seriam múltiplas mas ainda de formato análogo, apenas tendo uma correspondente para cada símbolo de  $L$ .

Descrevendo de forma direta, a função de contagem que definimos faz uso de um *loop* (estrutura de repetição) e vai efetivamente **mapeando** cada ocorrência de  $x$  com ocorrências dos símbolos em  $L$ , nunca repetindo itens envolvidos entre pares.

Para isso, é feita uma marcação no formato:

Sendo  $x \in \Gamma$  | Marcamos um  $x$  qualquer o substituindo pelo respectivo  $x_{\#}$

Essa marcação sinaliza que esse símbolo deve ser ignorado nas iterações seguintes do loop.

Vamos então destrinchar o funcionamento de  $\psi$ , transição a transição, para verificarmos formalmente a computação das seguintes propriedades:

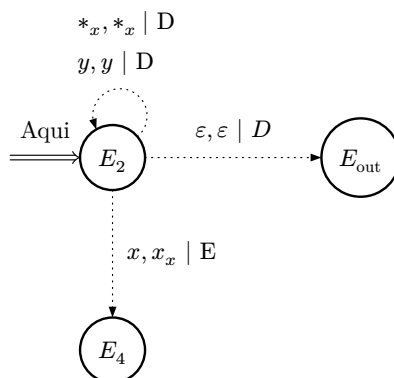
$$\text{Parar em } E_{>} \Leftrightarrow \gamma_x(w) > \gamma_y(w) + \gamma_z(w)$$

$$\text{Sair por } E_{\leq} \Leftrightarrow \gamma_x(w) \leq \gamma_y(w) + \gamma_z(w)$$

Passo 1 - Rebobina ao início da fita.

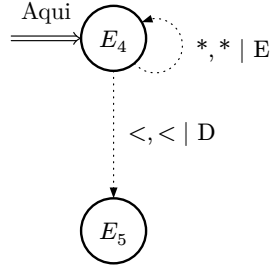


Passo 2 - Avança pela fita passando por todos  $y$  e  $x_x$  – que são os  $x$ 's já pareados – até encontrar um  $x$  ou o fim da fita.

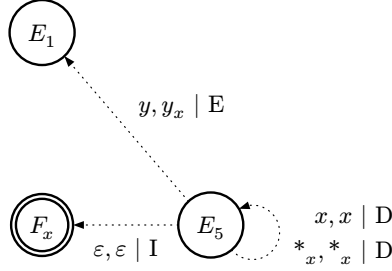


- a) Caso ache um  $x$  ou  $x_y$ , o sobreescreveremos por  $x_x$  e seguimos para o passo 3.
- b) Caso não ache um  $x$ , escapamos para  $E_{\leq}$ .

Passo 3 - Rebobina pro início da fita;



Passo 4 - Avança pela fita buscando um  $y$  tal que  $y \notin \{x, x_x\}$ ;



- a) Caso não encontre, pára em  $E_>$ .
- b) Caso encontre, sobreescreveremos o  $y$  na fita com  $y_x$  e retornamos ao passo 1.

Um aspecto relevante é que trata-se um *loop* com apenas duas possíveis interrupções: as etapas 2.b) e 4.a) do passo-a-passo.

Agora faremos uma breve verificação formal de que o algoritmo abstraído na máquina constrói as propriedades almejadas.

Dado que,

$$n, i \in N$$

$$x, k_n \in \{a, b, c\}$$

$$w = k_1 * \dots * k_n$$

Os passos 2.a) e 4.b) marcam os elementos substituindo-os por  $x_\#$  e  $y_\#$ , respectivamente construindo:

$$X_w = \{j \mid \forall j \in N \text{ tal que } 0 \leq j \leq n \text{ e } k_j = x\} \text{ (Ocorrências de X)}$$

$$\overline{X}_w = \{j \mid \forall j \in N \text{ tal que } 0 \leq j \leq n \text{ e } k_j \neq x\} \text{ (Ocorrências de não-X)}$$

Como fazemos uma substituição 2.a), rebobinamos, fazemos uma substituição em 4.b) e reiniciamos o loop. Estamos sempre compondo  $X$  e  $\overline{X}$ , um por um, mantendo igualdade quanto a cardinalidade dos conjuntos. Incrementando primeiro  $X$  e então  $\overline{X}$ .

Dessa forma, independente da iteração do loop, no instante precedendo a execução do passo 2.a), temos que  $|X_w| = |\overline{X}_w| + 1$ . Até que executarmos 4.b), retornando a igualdade  $|X_w| = |\overline{X}_w|$ .

Ou seja, somente nos pontos de interrupção do loop ocorre disruptura permanente dessas igualdades.

Em 2.b), não encontramos mais  $x$  e escapamos para  $E_{\leq}$ , como interrompemos o processamento do módulo, não buscamos mais  $y$ , mas temos certeza da cardinalidade de  $x$ , naquele instante alcançamos que  $|X| = |\bar{X}|$ . Entretanto, como não buscamos mais  $y$ , podemos afirmar de maneira forte que  $\gamma(x)_w \leq \gamma(y)_w + \gamma(z)_w$  visto que podem ainda haver mais ocorrências de  $y$ .

Já na parada em 4.a), temos do passo 2.a) que  $|X_w| = |\bar{X}_w| + 1$  e interrompemos o processamento em  $E_{>}$ . Neste caso,  $\gamma(x)_w \geq \gamma(y)_w + 1$ , o que equivale a  $\gamma(x)_w > \gamma(y)_w$ .

### 2.2.2. Módulo $\alpha$

O módulo  $\alpha$  irá analisar se o [corolário  \$\Delta\$](#)  vale para alguma letra em  $\{a, b, c\}$ . Caso verifique validade de tal propriedade para a letra-alvo, é eleito o estado final apropriado representativo da decisão apropriada e a computação cessa.

Visto que o automôto encerra processamento assim que é validado  $\Delta$  para um elemento, extraímos uma capacidade de “curto-circuito”, a qual em conjunto a um processamento ordenado dos elementos de  $\Sigma$  permite zelar pela regra de desempate definida em [1.2](#)

Portanto, efetuamos essa avaliação para as letra na ordem “a”, “b” e, então, “c”, implicitamente zelando pela ordem de desempate.

Caso nenhuma letra do alfabeto cumpra com a propriedade alvo de  $\alpha$ , a computação é encaminhada para o módulo  $\beta$ .

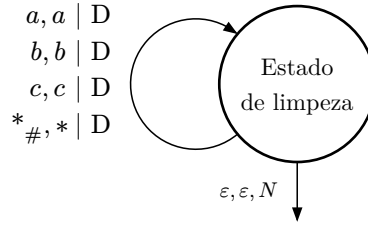
Vamos então quebrar o processamento em 3 instâncias de submódulos  $\psi$ , tratando os estados  $E_{>}$  de cada como a decisão de concessão de prioridade para a letra.

Esses submódulos serão intercalados por um estado de “limpeza” com transições que irão remover os símbolo marcados retornando o conteúdo da fita a entrada inicial. Dessa forma símbolos escritos por  $\psi$  não afetarão os submódulos seguintes.

Vale notar que existe uma alternativa que reduziria o custo computacional da MT, por meio da remoção desses estados de limpeza, incluindo ao alfabeto da fita símbolos de marcação específicos pra cada  $\psi$ , e então incorporando-os nas transições dos submódulos de contagem seguintes.

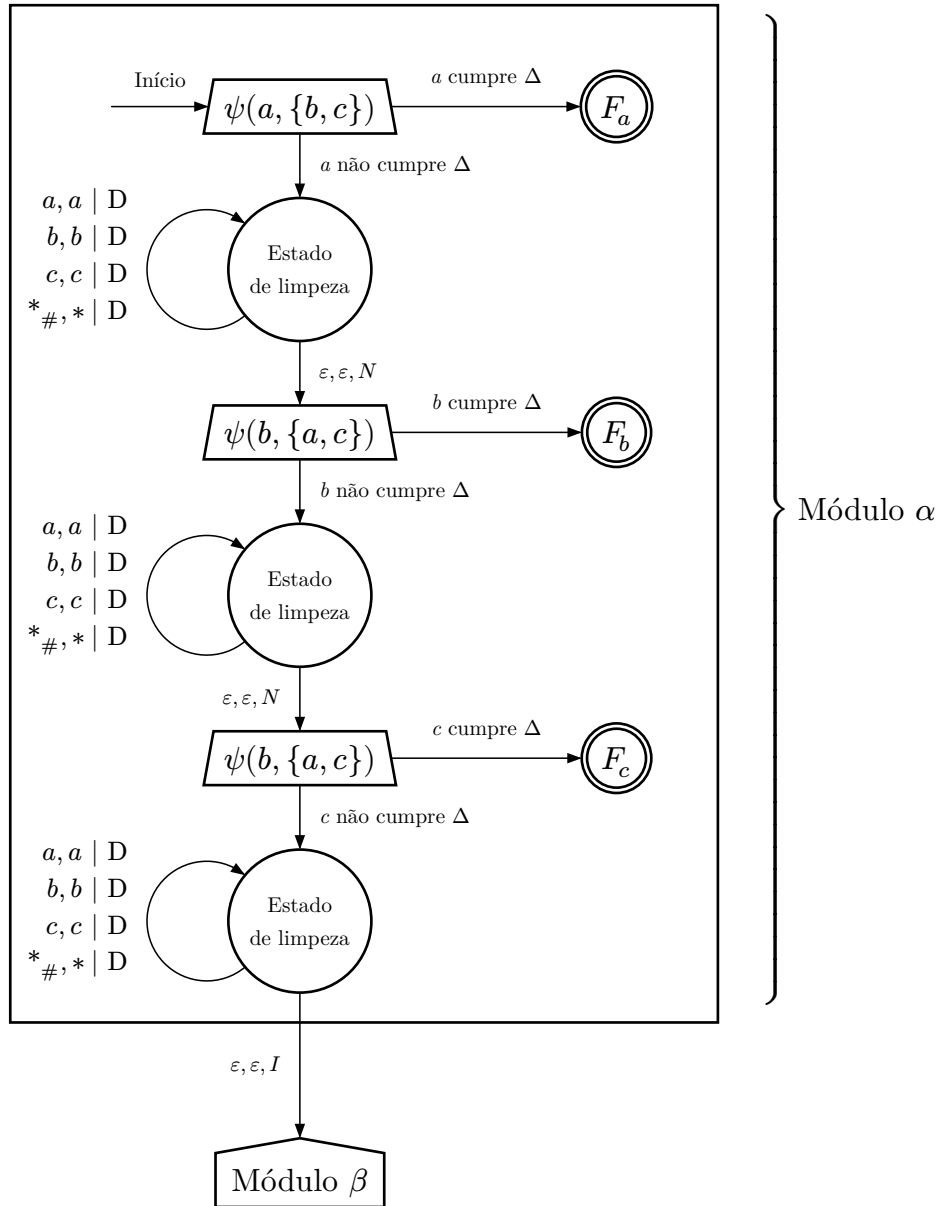
Apesar de mais eficiente, isso acrescentaria um número considerável de transições, então optamos por manter essa estrutura.

Cada estado de limpeza, segue o formato



Máquina de Turing 1: Estado de limpeza

Dando ao módulo  $\alpha$  o formato:



Fluxograma 2: Raio-x do módulo  $\alpha$

### 2.2.3. Do custo da contagem no módulo $\alpha$

$\psi(x, \{y, z\})$  deve verificar se para um certo  $x$  vale:

$$\gamma_x(w) > \gamma_y(w) + \gamma_z(w) \text{ onde } x, y, z \in \{a, b, c\} \text{ e } x \neq y \neq z$$

Para tal, queremos comparar a **cardinalidade** de  $x$  com a de  $y$  unida a de  $z$ . Com esse objetivo, utilizamos nossa função de comparação de cardinalidade  $\psi(x, L)$ , sendo  $L = \{y, z\}$ .

Dentro do módulo, substituímos o  $E_{>}$  por  $F_x$  simbolizando a parada total da MT1 na decisão de priorizar a avenida  $x$ . Uma vez que esse estado implica na condição verdade de  $\Delta$  para o  $x$  alvo.

Enquanto  $E_{\leq}$  será:

- $E_{\text{in}}$  do submódulo  $\psi$  seguinte, caso  $x \neq c$ ;
- o estado de entrada do módulo  $\beta$ , caso  $x = c$ ;

Dado a definição de  $x$  e  $L$ , vamos agora para o cálculo da complexidade temporal associado a  $\alpha$ .

Por convenção, nos preocupamos nessa análise com o pior caso possível, com isso em mente temos de determinar qual seria a palavra de maior custo associado para que  $\alpha$  compute.

Com esse fim, vamos destrinchar cada passo de  $\psi$ , determinando o pior formato possível da palavra para cada um deles. Depois, comparar essas composições de  $w$ , calcular o custo associado e eleger o pior candidato entre eles.

Trataremos por  $i$ , a iteração atual do passo-a-passo, visto que  $\psi$  opera como *loop*, iniciando nossa contagem em 1.

#### Passo 1 - Rebobinação da fita

— Pior palavra:  $w = x^j * y^k$

A partir de  $i > 0$ , essa etapa rebobina a partir da posição do cabeçote determinada no passo 4.b). O pior caso então é o de  $w$  tendo  $y^k$  como sufixo da palavra, maximizando a quantidade de transições para rebobinar.

#### Passo 2 - Busca por um $x$ não marcado

— Pior palavra:  $w = y^k * x^j$

Como aqui buscamos um  $x$  não pareado, iniciando nossa busca do início de  $w$  seguindo linearmente. O pior caso é de todos  $x^j$  como sufixo de  $w$ .

#### Passo 3 - Rebobinação da fita

— Pior palavra:  $w = y^k * x^j$

Análogo ao primeiro passo, só que dessa vez a posição inicial do cabeçote é determinada pelo passo 2), assim invertemos a ordem.

#### Passo 4 - Buscamos por um $y$ não marcado

— Pior palavra:  $w = x^j * y^k$

Análogo ao passo 2), só que dessa vez queremos que o sufixo de  $w$  seja  $y^k$ .

Temos então dois formatos possíveis de  $w$  disputando a posição de pior possível. Vamos considerar cada um e calcular o custo associado.

Considere a seguir que,

$$w \text{ é palavra e, } n = |w| = m + k$$

#### 2.2.3.1. Análise para $w = y^k * x^m$

- **Passo 1:** Da posição  $i$  até o início  $\rightarrow i$
- **Passo 2:** Passar por  $k$  símbolos  $y$  +  $i$  símbolos  $x_{\#} \rightarrow k + i$
- **Passo 3:** Da posição  $k + i$  até o início  $\rightarrow k + i$
- **Passo 4:** Passar pelos primeiros  $i$  símbolos até o próximo  $y$  livre  $\rightarrow i$

– **Custo por iteração ( $C_i$ )**

$$C_i = i + (k + i) + (k + i) + i = 2k + 4i$$

#### 2.2.3.2. Análise para $w = x^m y^k$

- **Passo 1:** Da posição  $m + i$  até o início  $\rightarrow m + i$
- **Passo 2:** Passar por  $i$  símbolos  $x_{\#}$  até o próximo  $x \rightarrow i$
- **Passo 3:** Da posição  $i$  até o início  $\rightarrow i$
- **Passo 4:** Passar por  $m$  símbolos  $x$  (todos já marcados) +  $i$  símbolos  $y_{\#} \rightarrow m + i$

**Custo da iteração  $i$ :**

$$C_i = (m + i) + i + i + (m + i) = 2m + 4i$$

Para  $y^k x^m$  com  $\ell$  iterações:

$$T_1 = \sum_{i=1}^{\ell-1} (2k + 4i) = 2k\ell + 2\ell(\ell - 1)$$

Para  $x^m y^k$  com  $\ell$  iterações:

$$T_2 = \sum_{i=1}^{\ell-1} (2m + 4i) = 2m\ell + 2\ell(\ell - 1)$$

Resta determinar para cada caso a pior distribuição de  $m$  e  $k$ .

Em  $T_1$ ,  $\ell$  é o número de iterações do *loop*, sendo esse determinado de acordo aos dois cenários de parada.

- Caso 1:  $k \leq m \rightarrow$  Pára em  $F_x \Leftrightarrow \ell = k$
- Caso 2:  $k > m \rightarrow$  Escapa para  $E_{\text{out}} \Leftrightarrow \ell = m$

Portanto,  $\ell = \min(k, m)$ .

Substituindo em  $T_1$ ,

$$T_1(k) = 2k \cdot \min(k, m) + 2 \min(k, m) \cdot (\min(k, m) - 1)$$

$$T_1(k) = 2k \cdot \min(k, m) \cdot [k + \min(k, m) - 1]$$

- Caso 1:  $\min(k, m) = k$

$$T_1(k) = 2k \cdot k + 2k \cdot (k - 1) = 4k^2 - 2k$$

Com a restrição de  $k + m = n$  e  $k \leq m$ :

$$k \leq m \implies k \leq n - k \implies 2k \leq n \implies k \leq \frac{n}{2}$$

Importante notar que estamos considerando  $|w| > 0$  aqui (ou seja, uma palavra não vazia), logo  $n \geq 2$ .

Com isso temos o domínio  $k \in [1, \frac{n}{2}]$ .

Vamos otimizar,

$$\frac{dT_1}{dk} = 8k - 2$$

Em nosso domínio obtemos que, seja  $k \geq 1$ ,

$$8k - 2 \geq 8(1) - 2 = 6 > 0$$

Como nossa função é estritamente crescente no intervalo relevante, basta pegarmos o extremo da direita para termos o máximo.

Ou seja, no caso 1), a pior distribuição possível é  $k = \frac{n}{2}$ , e como  $n = k + m$ , assim obtemos  $k = m = \frac{n}{2}$ .

O que nos dá um custo máximo de:  $T_1^A = 4(\frac{n}{2})^2 - 2(\frac{n}{2}) = n^2 - n$

- Caso 2:  $\min(k, m) = m = n - k$

$$T_1(k) = 2k \cdot (n - k) + 2(n - k) \cdot ((n - k) - 1) = 2(n - k)(k - 1)$$

$$T_1(k) = 2(n - 1)(n - k) = 2n^2 - 2n - 2nk + 2k$$

Aqui o domínio é  $m \in (\frac{n}{2}, n)$ , visto que  $k > m = n - k$ .

Derivando,

$$\frac{dT_1}{dk} = -2n + 2 = 2(1 - n)$$

Para  $n > 1$ ,  $2(1 - n) < 0$ , ou seja, a função é decrescente no domínio inteiro. Logo basta pegarmos o valor a extrema esquerda para  $k = (\frac{n}{2})$ .

O custo desse caso será  $T_1^B = 2(\frac{n}{2})(n - 1) = n(n - 1) = n^2 - n$ . O mesmo do caso A).

Ou seja, o custo computacional é o mesmo independente de qual caso de interrupção, e a pior composição possível para a palavra dado o formato  $w = y^k + x^m$  é  $k = m = \frac{|w|}{2}$ .

Outro fato importante que extraímos da análise da derivada, é de que a função é crescente a medida que os valores de  $k$  e  $m$  convergem.

Por fim, determinamos o custo de um  $\psi$  em  $\alpha$  igual à  $n^2 - n$ .

Em  $T_2$ , por simetria, a análise é idêntica e o mesmo custo obtido.

Determinamos a pior palavra para qualquer  $\psi$  em  $\alpha$  sendo:

$$x, y, z \in \{a, b, c\}$$

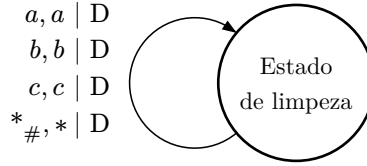
$$w = x^i \cdot y^i \cdot z^i$$

com qualquer permutação de  $x$ ,  $y$  e  $z$ .

Obtendo custo total de  $n^2 - n$ , ou seja, contido no  $\Theta(n^2)$ .

#### 2.2.4. O custo da limpeza

Vamos fazer uma breve demonstração da complexidade da operação de limpeza, previamente definidas como:



Máquina de Turing 2: Estado de limpeza

O custo aqui é o da distância do ponto de entrada no *estado de limpeza* ao fim da palavra de entrada, visto que iteramos por todos os símbolos removendo os marcados.

Como entramos nesse estado somente no caso de escape de  $\psi$  por  $E_{\leq}$  que sempre precede o posicionamento do cabeçote no início da fita.

Ou seja, o cabeçote estará no início da fita, portanto o custo aqui é de  $n$ , sendo  $n = |w|$ , o qual obviamente pertence a  $\Theta(n)$ .

#### 2.2.5. Conclusão do custo de $\alpha$

Podemos finalmente estipular o custo acumulado de  $\alpha$ . Dado que consideramos o pior caso como o de empate contínuo, teremos:

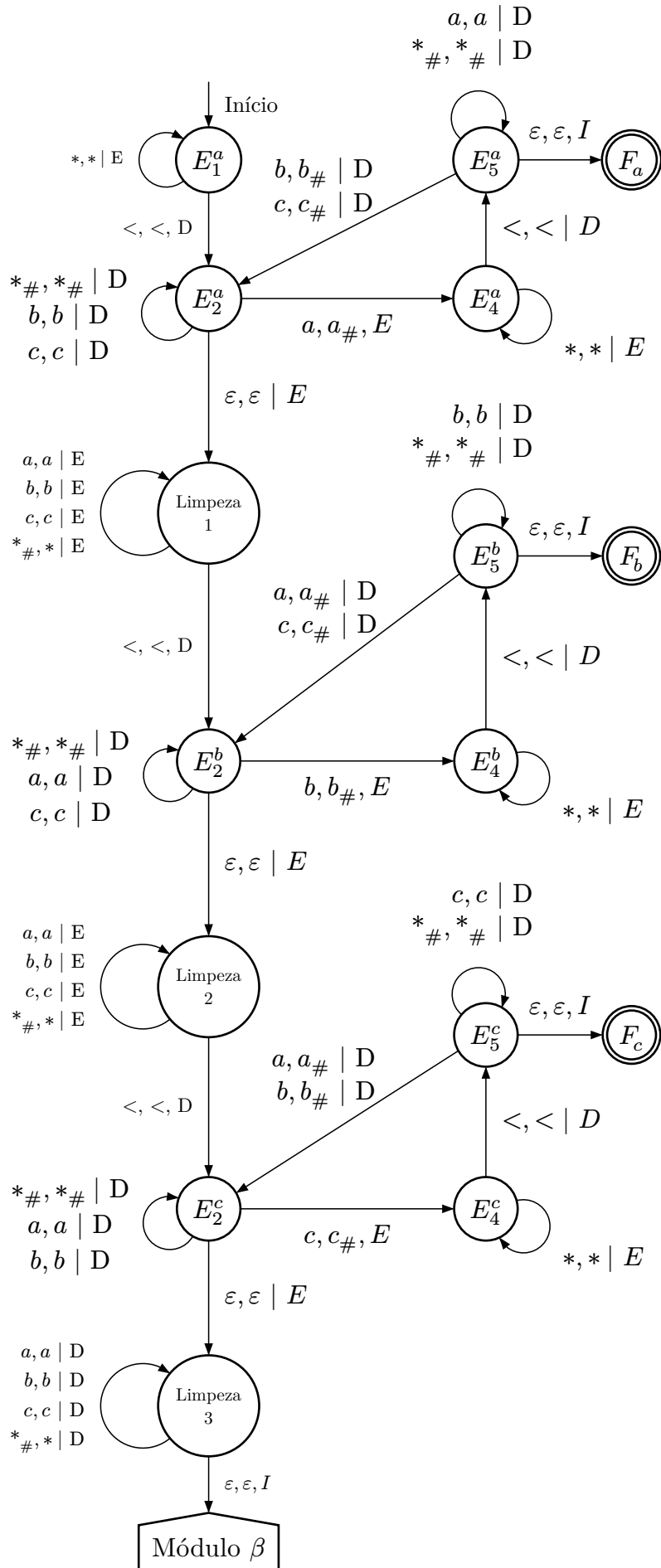
1.  $\psi(a, \{b, c\})$ :  $n^2 - n$
2. Operação de limpeza:  $n$
3.  $\psi(b, \{a, c\})$ :  $n^2 - n$
4. Operação de limpeza:  $n$
5.  $\psi(c, \{a, b\})$ :  $n^2 - n$
6. Operação de limpeza:  $n$

Totalizando,

$$\text{Custo total de } \alpha = 3((n^2 - n) + n) = 3n^2$$



### 2.2.6. Módulo $\alpha$ na íntegra



Módulo 1: Módulo  $\alpha$

### 2.2.7. Módulo $\beta$

O módulo  $\beta$  deve avaliar a condição de desempate, a qual trataremos de  $\Omega$  e a ordem de prioridade  $A > B > C$ .

Afirmamos que  $x$  cumpre  $\Omega$  quando:

$$x, y, z = \{a, b, c\}, x \neq y \wedge y \neq z \mid \max(\gamma_x(w), \gamma_y(w), \gamma_z(w)) = \gamma_x(w)$$

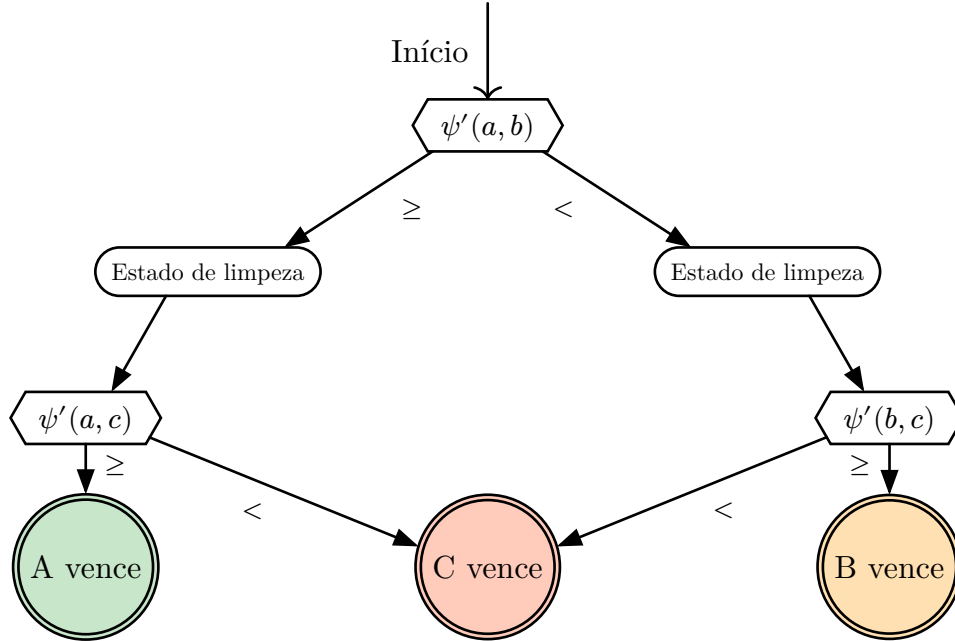
Ou seja,

$$\gamma_x(w) > \gamma_y(w) \wedge \gamma_x(w) > \gamma_z(w)$$

Entranto, na condição de haver um empate, onde dois símbolos possuam a mesma cardinalidade, o desempate será definido pela ordem de prioridade.

Para determinar e resolver casos de empate com maior facilidade, vamos fazer uma pequena modificação em  $\psi$ , para que ao invés de termos como possíveis estados de saída  $E_{>}$  e  $E_{\leq}$ , tenhamos agora  $E_{\geq}$ ,  $E_{<}$  para satisfazer a estrutura que propomos aqui. A essa nova versão de  $\psi$  daremos o nome  $\psi'$ .

Para que então possamos estruturar o módulo  $\beta$  como um torneio de entre as três letras de  $\{a, b, c\}$ , onde cada confronto compara as cardinalidades de dois símbolos e decide o próximo embate. O formato definido segue:

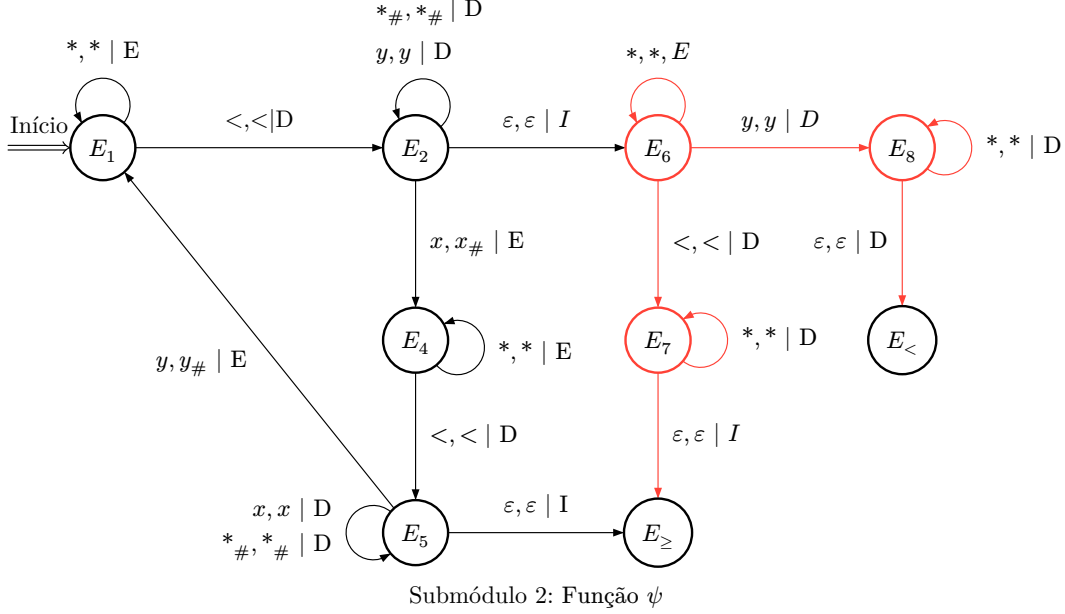


Fluxograma 3: Ordem dos duelos do Torneio  $\beta$

### 2.2.8. Função $\psi'$

Vamos fazer os ajustes necessário em  $\psi$  para obter dois novos estados de escape  $E_{\geq}$  e  $E_{<}$ , respectivamente, representando “maior ou igual a” e “menor que”.

Para tal, definimos  $\psi'$  como segue:



Preservamos todas as transições de  $\psi$ , apenas substituímos  $E_{>}$  por  $E_{\geq}$  e  $E_{\leq}$  por  $E_{<}$ , e acrescentamos um loop de busca por um  $x$ .

De forma que, após determinarmos que não existem mais símbolos  $x$  na fita após  $E_2$ , buscamos por pelo menos um símbolo  $y$  para garantir que  $\gamma_x(w) < \gamma_y(w)$ .

Sabemos, com base na prova da certeza de  $\psi$  que em  $E_2$ ,  $|X| = |\overline{X}|$ .

Vamos usar dessa hipótese e tratar as novas transições e estados – que estão destacados em vermelho – como uma MT isolada e demonstrar como ela apropriadamente determina se  $|X| < |Y|$  ou  $|X| \geq |Y|$ .

#### 2.2.8.1. Prova da certeza de $\psi'$

Vamos verificar se a submáquina formada por  $\{E_6, E_7, E_8, E_{\geq}, E_{<}\}$  determina se  $|X| < |Y|$  ou  $|X| \geq |Y|$  corretamente:

Temos que nos estados,

- $E_6$ : busca por símbolos  $y$  não processados
- $E_7$ : encontrou pelo menos um  $y$  restante
- $E_8$ : verifica ausência de  $y$  restantes
- $E_{\geq}$ : estado final para  $|X| \geq |Y|$
- $E_{<}$ : estado final para  $|X| < |Y|$

Tome nota desses dois estados relevantes:

1. Em  $E_2$ , temos:

$$E_2 \xrightarrow{\varepsilon, \varepsilon \mid I} E_6$$

2. Em  $E_6$ , a máquina move-se à direita até:

$$E_6 \xrightarrow{y, y \mid D} E_7 \quad \text{ou} \quad E_6 \xrightarrow{<, < \mid D} E_8$$

Então, dado uma palavra  $w$  na fita e os conjuntos de ocorrência  $X$  e  $\overline{X}$ , que foram previamente definidos, vamos supor os 3 possíveis casos quanta a comparação da cardinalidade:

**Caso 1:**  $|X| = |Y|$

Teremos todos os  $x$  e  $y$  pareados já pareados, vide o funcionamento da  $\psi$ . A máquina não encontra  $y$  restantes em  $E_6$ , eventualmente alcançando  $E_{\geq}$  (diretamente ou via  $E_8$ ).

**Caso 2:**  $|X| < |Y|$

Existirão símbolos  $y$  não processados, vide o funcionamento de  $\psi$ . A execução segue:

$$E_6 \xrightarrow{y, y \mid D} E_8 \xrightarrow{*, * \mid D} E_8 \xrightarrow{\varepsilon, \varepsilon \mid I} E_{<}$$

**Caso 3:**  $|X| > |Y|$

Este caso é **impossível** ao alcançar  $E_2$ . A transição  $E_2 \xrightarrow{\varepsilon, \varepsilon \mid I} E_6$  só ocorre quando não há mais símbolos  $x$  na fita. Pela construção de  $\psi$ , cada  $x$  é marcado como  $x_{\#}$  somente quando há símbolo correspondente disponível. Se  $|X| > |Y|$ , a máquina já teria alcançado o estado  $E_{\geq}$  por  $E_5$ .

Temos então certeza de que o módulo definido pelos estados  $\{E_6, E_7, E_8, E_{\geq}, E_{<}\}$  e suas respectivas transições computa corretamente a regra:

$$\begin{cases} \text{aceita em } E_{\geq} & \text{se } |X| \geq |Y| \\ \text{aceita em } E_{<} & \text{se } |X| < |Y| \end{cases}$$

### 2.2.8.2. Análise temporal de $\psi'$

$\psi'$  conserva o mesmo conjunto de transições  $\psi$ , apenas acrescentando um último loop de busca e rebobinação para achar um  $x$ , dessa forma, podemos recorrer aos resultados obtidos na análise temporal anterior, e apenas incrementar o custo associado a essa nova etapa para entender o novo pior caso.

Antes, primeiro vamos rapidamente demarcar **qual é o novo pior caso** para isso vamos considerar os três possíveis cenários que culminem em escape:

$$1. E_5 \xrightarrow{\varepsilon, \varepsilon | I} E_{\geq}$$

Aqui temos o resultado de  $|X| \geq |\bar{X}|$  como já demonstrado na análise de  $\psi$ . Aqui não há nenhum acrescimo de custo a complexidade já calculada.

$$2. E_7 \xrightarrow{\varepsilon, \varepsilon | I} E_{\geq}$$

O caso em que  $|X| > |\bar{X}|$ , percorrá o trajeto inteiro de:

$$E_6 \xrightarrow{*,* | E} E_6 \xrightarrow{<, < | D} E_7 \xrightarrow{*,* | D} E_7 \xrightarrow{\varepsilon, \varepsilon | I} E_{\geq}$$

Iniciando com o cabeçote no fim da palavra, as transições produzirão um custo, em ordem, de:  $n, 1, n, 0$ . Acumulando um adicional de  $2n + 1$ . Interessante notar que aqui é indiferente a distribuição das letras na palavra, já que todos os símbolos relevantes já teriam sido pareados.

$$3. E_8 \xrightarrow{\varepsilon, \varepsilon | I} E_{<}$$

O caso onde  $|X| < |\bar{X}|$ , percorrá o trajeto inteiro de:

$$E_6 \xrightarrow{*,* | E} E_6 \xrightarrow{y, y | D} E_8 \xrightarrow{*,* | D} E_8 \xrightarrow{\varepsilon, \varepsilon | I} E_{<}$$

De forma análoga ao outro módulo, a ordem da composição da palavra traz se um aumento de custo desde que haja uma formação homogênea do estilo  $w = x^j + y^k + z^l$ , com o  $x$ ,  $y$  e  $z$  mais próximos possíveis. Como faremos duelos envolvendo todos os símbolos, a permutação específica das sequências não altera o custo total. A pior palavra seria:  $w = x^m * y^k * z^j$  onde  $m < k + j$ .

Obtemos o custo por iteração (denotada por  $i$ ) de:

$$C_3 = n + (m + 1) + (m + 1) = n + 2m + 2$$

Como  $n = m + k + j$  e sabemos que  $m < k + j$ , o pior caso é maximizar  $m$  respeitando essa restrição, vide análise da derivada da equação do custo temporal de  $\psi$ .

$$n = m + (m + 1) = 2m + 1$$

Substituindo em  $C_3$ :

$$C_3 = (2m + 1) + 2m + 2 = 4m + 3$$

Ou em termos de  $n$ :

$$m = \frac{n-1}{2}$$

$$C_3 = 4 \cdot \frac{n-1}{2} + 3 = 2n - 2 + 3 = 2n + 1$$

Obtendo um acrescimo total de  $2n + 1$ .

Dessa forma, o custo associado  $\psi'$ , considerando o caso 1 ou 3, é de

$$\text{custo de } \psi' = \text{custo adicional} + \text{custo de } \psi = (2n + 1) + (n^2 - n) = n^2 + n + 1$$

que também pertence a  $\Theta(n^2)$ .

**Portanto, os casos 1 e 3 tem o mesmo custo, apesar de implicarem em diferentes distribuições para os símbolos.**

Resta então considerarmos a estrutura maior do torneio para eleger a pior palavra. Como o vencedor participará de 2 duelos, ele deve ser o sufixo mais distante do início da palavra, maximizando o custo associado a computação total, as demais sequências são intercambiáveis.

Então para um  $X$  vencedor, a palavra de entrada que maximiza o custo temporal é:  $w = z^j + y^k + x^m$ , onde  $j = k = m - 1$ . Implicando num custo total de

$$C_{\text{total}} = 2 \cdot \text{custo da limpeza} + 2 \cdot \text{custo por duelo}$$

$$C_{\text{total}} = 2 \cdot (n) + 2 \cdot (n^2 + n + 1) = 2n + 2n^2 + 2n + 2$$

$$C_{\text{total}} = 2n^2 + 4n + 2$$

que também é  $\Theta(n^2)$ .

### 2.2.8.3. Custo total da $MT_1$

Portanto, no pior caso,  $MT_1$ , recebe como entrada:

$$w = z^j + y^k + x^m \vee w = y^j + z^k + x^m$$

Sendo,

$$k = j = m - 1 \wedge x, y, z \in \{a, b, c\} \wedge x \neq y \wedge y \neq z$$

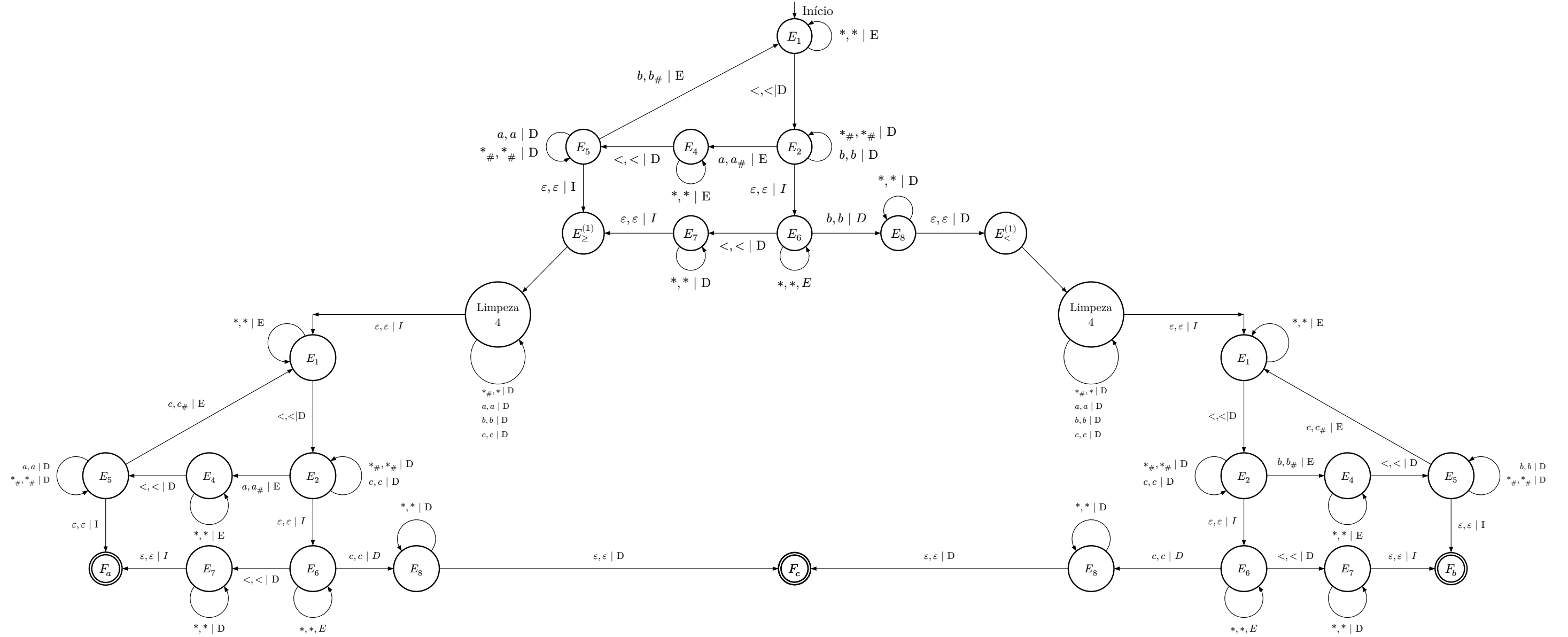
E determina a avenida  $X$  como prioritária com o custo computacional total de:

$$T(n) = T_{\text{alfa}} + T_{\text{beta}} = (3n^2) + (2n^2 + 4n + 2) = 5n^2 + 4n + 2$$

Logo,

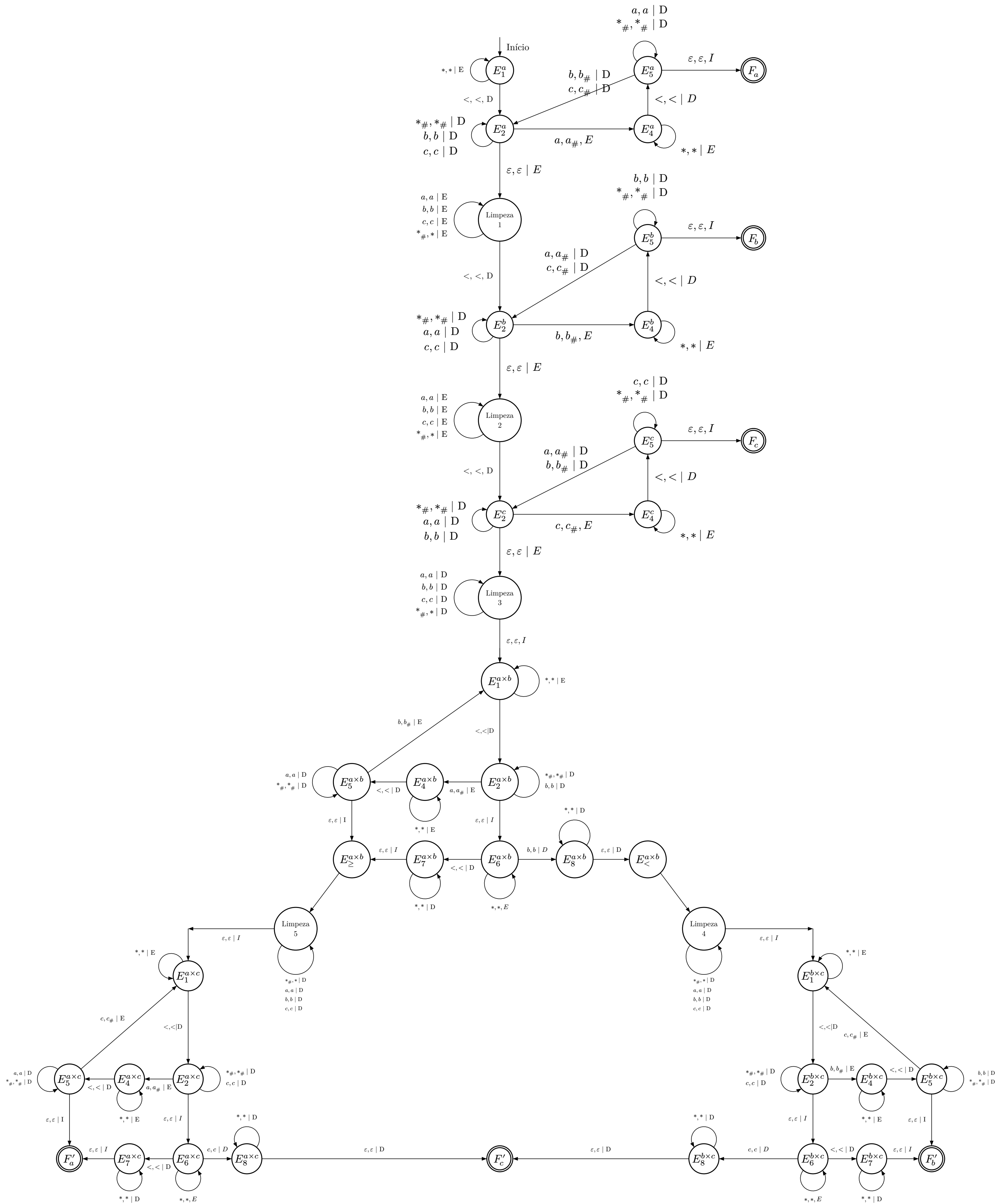
$$MT_1 \in \Theta(n^2)$$

### 2.2.9. Módulo $\beta$ na íntegra



Módulo 2: Módulo  $\beta$

### 2.3. Máquina na íntegra

Módulo 3: Módulos  $\alpha$  e  $\beta$



## 2.4. Exemplos passo a passo

### 2.4.1. Exemplo 1

Considere a entrada:

$$w = A^4 B^2 C^1$$

Já na avaliação da primeira função de contagem,  $\psi(a, \{b, c\})$ , já é determinado a superioridade da cardinalidade de  $a$  sobre a soma das outras duas, logo ocorre parada em  $F_a$  e a avenida  $A$  é definida como prioritária.

O processo de computação sendo composto apenas pelos passos do primeiro submódulo de contagem até o esgotamento dos símbolos  $b$  e  $c$ , culminando num  $a$  não pareado restante na fita.

### 2.4.2. Exemplo 2

Considere a entrada:

$$w = C^3 A^3 B^3$$

Aqui a computação será mais demorada, apesar da mesma tomada de decisão final – a da priorização da avenida  $A$  –, existe um empate na cardinalidade dos símbolos, o que exige a avaliação do módulo  $\beta$  para desempate.

Portanto, serão feitas todas as comparações de  $\alpha$ , até que seja verificado que:

$$\gamma_w(a) \not\geq \gamma_w(b) \wedge \gamma_w(b) \not\geq \gamma_w(a) \wedge \gamma_w(b) \not\geq \gamma_w(c) \wedge \gamma_w(c) \not\geq \gamma_w(a)$$

Ou seja, não há símbolo com cardinalidade maior que os outros, e o módulo  $\beta$  é acionado para desempate.

Lá serão computados os duelos  $A \times B$  e  $A \times C$ , ambos determinando empate e seguindo ao estado final  $F'_a$

## 3. Máquina de múltiplas fitas (MT<sub>2</sub>)

Com o objetivo de apresentar uma solução mais eficiente do que a MT1, introduzimos uma Máquina de Turing determinística de múltiplas fitas, denominada MT2.

A principal motivação para o uso de múltiplas fitas é permitir uma separação clara entre as etapas de leitura, contagem, comparação e decisão. Essa separação elimina a necessidade de rebobinamentos frequentes da fita de entrada, reduzindo significativamente o custo computacional total do algoritmo, sem comprometer a corretude da solução.

Assim como a MT1, a MT2 resolve o problema descrito no enunciado: determinar qual avenida deve receber o sinal verde em um cruzamento urbano, a partir da quantidade de veículos em cada via, respeitando a política de decisão e a ordem de desempate  $A > B > C$ .

### 3.1. Definição formal e organização em módulos

A MT2 é definida como uma Máquina de Turing determinística de múltiplas fitas, representada pela seguinte 7-upla:

$$MT2 = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{final}})$$

onde:

- $Q$  é um conjunto finito de estados;
- $\Sigma = \{A, B, C\}$  é o alfabeto de entrada;
- $\Gamma = \{A, B, C, \_, \}$  é o alfabeto da fita, em que  $\_$  representa o símbolo vazio;
- $\delta$  é a função de transição determinística da máquina;
- $q_0$  é o estado inicial;
- $q_{\text{final}}$  é o estado final de aceitação.

A função de transição possui a forma:

$$\delta : Q \times \Gamma^4 \rightarrow Q \times \Gamma^4 \times \{E, D, N\}^4$$

atuando simultaneamente sobre quatro fitas: uma fita de entrada e três fitas auxiliares. Em cada transição, a máquina lê um símbolo de cada fita, escreve um símbolo em cada fita e move cada cabeçote de forma independente, garantindo determinismo durante todo o processamento.

A MT2 opera com quatro fitas: uma fita de entrada e três fitas auxiliares. A fita de entrada contém a palavra que representa a distribuição de veículos, enquanto as fitas auxiliares armazenam, em notação unária, as quantidades de veículos associadas a cada avenida.

A palavra de entrada segue o formato:

$$A^i B^j C^k$$

em que  $i$ ,  $j$  e  $k$  representam, respectivamente, o número de veículos nas avenidas A, B e C.

Para adequar o enunciado original do problema ao modelo formal de Máquinas de Turing, as quantidades numéricas de veículos são representadas na fita de entrada por meio de codificação unária. Assim, cada ocorrência do símbolo  $A$  representa um veículo na avenida A, cada  $B$  representa um veículo na avenida B, e cada  $C$  representa um veículo na avenida C, resultando em uma correspondência direta entre os valores do enunciado e a palavra processada pela máquina.

O processamento da MT2 é organizado nos seguintes módulos lógicos:

- leitura da entrada e contagem das ocorrências de cada símbolo;
- cálculo do número total de veículos;
- cálculo da metade do total;
- verificação de maioria absoluta;
- comparação das quantidades em caso de ausência de maioria;

- escrita da decisão final.

Essa organização modular facilita a compreensão do funcionamento da máquina e permite uma análise de complexidade estruturada.

### 3.2. Representação das fitas

Durante a execução da MT2, cada fita possui uma função bem definida. A primeira fita contém a palavra de entrada e é utilizada exclusivamente na etapa inicial de leitura. As três fitas auxiliares armazenam, em notação unária, o número de ocorrências dos símbolos  $A$ ,  $B$  e  $C$ , respectivamente.

Após a fase de leitura, não é necessário retornar à fita de entrada, pois todas as informações relevantes para a decisão já se encontram armazenadas nas fitas auxiliares. Esse aspecto é fundamental para a obtenção de complexidade linear.

### 3.3. Funcionamento da máquina

Inicialmente, a MT2 percorre a fita de entrada da esquerda para a direita. Durante essa varredura, cada ocorrência dos símbolos  $A$ ,  $B$  e  $C$  é registrada na fita auxiliar correspondente, por meio da escrita de um símbolo 1 para cada ocorrência.

Concluída a leitura, a máquina constrói uma representação unária do número total de veículos no cruzamento sobrescrevendo a fita de entrada, contendo exatamente  $i + j + k$  símbolos 1.

Em seguida, a MT2 calcula a metade desse total. Para isso, dois símbolos 1 são removidos por vez da fita do total, e um símbolo 1 é escrito em uma fita auxiliar para cada par removido. Caso reste um símbolo sem par, este é descartado, obtendo-se assim a representação unária de:

$$\left\lceil \frac{i + j + k}{2} \right\rceil$$

Com o valor da metade calculado, a MT2 compara as quantidades de veículos de cada avenida com esse valor. As comparações são realizadas de forma sequencial, seguindo a ordem A, B e C, por meio de varredura simultânea e não destrutiva das fitas envolvidas, sendo encerradas quando uma das fitas atinge o símbolo branco  $\_$ .

Se alguma avenida possuir uma quantidade de veículos estritamente maior que a metade do total, essa avenida é imediatamente definida como prioritária, e o símbolo correspondente pertencente a  $\{A, B, C\}$  é escrito na fita de saída. Após o registro da decisão, o processamento é encerrado, e a máquina transita para o estado  $q_{\text{final}}$ .

Caso nenhuma avenida apresente maioria absoluta, a MT2 inicia uma nova fase de decisão baseada em comparações diretas entre as quantidades individuais armazenadas nas fitas auxiliares das avenidas.

Essas comparações são realizadas duas a duas, por meio de varredura simultânea das fitas correspondentes, sem modificação de seus conteúdos. Em cada comparação, os cabeçotes avançam sincronamente até que uma das fitas atinja o símbolo branco  $\_$ .

A fita que alcançar o símbolo  $\_$  primeiro é considerada como representando a menor quantidade.

Inicialmente, a MT2 compara as fitas das avenidas A e B. Se a fita da avenida A for estritamente maior que a da avenida B, o símbolo A é imediatamente escrito na fita de saída como candidato prioritário. Caso contrário, o símbolo B é escrito na fita de saída.

Em seguida, a avenida cujo símbolo encontra-se registrado na fita de saída é comparada com a avenida C, utilizando o mesmo procedimento. Se a quantidade da avenida registrada for estritamente maior, seu símbolo é mantido na fita de saída. Caso contrário, o símbolo C sobrescreve o conteúdo da fita de saída.

Se, durante qualquer uma dessas comparações, as duas fitas atingirem o símbolo branco  $\_$  simultaneamente, caracteriza-se um empate. Nessa situação, a MT2 resolve a decisão com base no critério de prioridade fixo estabelecido no projeto da máquina, dado pela ordem:

$$A > B > C$$

Nessa situação, o símbolo correspondente à avenida prioritária segundo essa ordem é escrito na fita de saída.

Ao final do processamento, a fita de saída contém exatamente um único símbolo pertencente ao conjunto  $\{A, B, C\}$ , correspondente à avenida definida como prioritária. A execução é encerrada ao alcançar o estado  $q_{\text{final}}$ .

### 3.4. Análise formal de complexidade

Seja  $w = A^i B^j C^k$  a palavra de entrada fornecida à MT2 e seja

$$n = |w| = i + j + k$$

o tamanho da entrada.

#### 3.4.1. Leitura da entrada e contagem

A MT2 percorre a fita de entrada uma única vez, da esquerda para a direita, até encontrar o símbolo branco  $\_$ .

Durante essa varredura:

- cada um dos  $n$  símbolos da entrada é lido exatamente uma vez;
- para cada símbolo lido, é realizada exatamente uma escrita de um símbolo 1 em uma das fitas auxiliares;
- não há rebobinamentos da fita de entrada.

Portanto:

$$T_1(n) = n$$

e

$$T_1(n) \in O(n)$$

### 3.4.2. Construção da fita do total de veículos

A construção da fita do total escreve exatamente  $n$  símbolos 1, logo:

$$T_2(n) = n$$

e

$$T_2(n) \in O(n)$$

### 3.4.3. Cálculo da metade do total

Para calcular a metade do total de veículos, a máquina remove dois símbolos 1 por iteração da fita do total e escreve um símbolo 1 na mesma fita, agora representando a metade.

No pior caso:

$$\left\lfloor \frac{n}{2} \right\rfloor$$

iterações são realizadas. Assim:

$$T_3(n) \in O(n)$$

### 3.4.4. Verificação de maioria absoluta e comparações

A verificação de maioria absoluta é realizada comparando, de forma sequencial, as quantidades de veículos de cada avenida com o valor da metade do total, seguindo a ordem A, B e C.

Cada comparação é feita de maneira não destrutiva. Para comparar a fita de uma avenida  $X \in \{A, B, C\}$  com a fita da metade, a MT2 posiciona simultaneamente os cabeçotes no início das duas fitas e realiza uma varredura síncrona da esquerda para a direita, avançando um símbolo por passo, sem realizar escritas.

A comparação é encerrada assim que um dos cabeçotes encontra o símbolo branco  $\_$ . Se a fita da metade atingir o símbolo  $\_$  primeiro, conclui-se que a avenida X possui uma quantidade de veículos estritamente maior que a metade do total. Caso a fita da avenida X atinja o símbolo  $\_$  antes (ou simultaneamente), conclui-se que não há maioria absoluta para essa avenida.

O número de passos executados em cada comparação é igual ao comprimento da menor fita entre a avenida considerada e a fita da metade. Assim, o custo de uma comparação envolvendo a avenida X é dado por:

$$T_X = \min\left(|X|, \left\lceil \frac{n}{2} \right\rceil\right)$$

No pior caso, nenhuma avenida possui maioria absoluta, e as três comparações são realizadas. Portanto, o custo total dessa etapa é:

$$T_4(n) = \sum_{X \in \{A, B, C\}} \min\left(|X|, \left\lceil \frac{n}{2} \right\rceil\right)$$

Como  $|A| + |B| + |C| = n$ , segue que:

$$T_4(n) \leq n$$

e, consequentemente:

$$T_4(n) \in O(n)$$

### 3.4.5. Escrita da saída e encerramento

Após a conclusão das etapas de verificação e comparação, a MT2 determina uma única avenida como prioritária.

Nessa fase final, a máquina:

- limpa a fita de saída;
- escreve exatamente um único símbolo pertencente ao conjunto  $\{A, B, C\}$ , correspondente à avenida selecionada;
- realiza a transição para o estado final de aceitação  $q_{\text{final}}$ .

Todas as operações dessa etapa envolvem um número constante de leituras, escritas e movimentos de cabeçote, independentemente do tamanho da entrada. Assim, o custo temporal dessa fase é constante:

$$T_5(n) \in O(1)$$

### 3.4.6. Custo total no pior caso

No pior caso, nenhuma avenida possui maioria absoluta, e todas as etapas do processamento da MT2 são executadas completamente.

Somando os custos das fases individuais, tem-se:

- leitura da entrada e contagem:

$$T_1(n) = n$$

- construção da fita do total:

$$T_2(n) = n$$

- cálculo da metade do total:

$$T_3(n) = \left\lfloor \frac{n}{2} \right\rfloor$$

- verificação e comparações:

$$T_4(n) \in O(n)$$

- escrita da saída e encerramento:

$$T_5(n) \in O(1)$$

Portanto, o custo total de execução no pior caso é dado por:

$$T(n) = n + n + \left\lfloor \frac{n}{2} \right\rfloor + O(n) + O(1)$$

Agrupando os termos dominantes, obtém-se:

$$T(n) \in O(n)$$

### 3.4.7. Complexidade em número de estados

A MT2 é definida por um conjunto finito de módulos lógicos bem delimitados: leitura e contagem, construção do total, cálculo da metade, verificação de maioria, comparações finais e escrita da saída.

Cada módulo é implementado por um número constante de estados, independente do tamanho da entrada ou dos valores  $i$ ,  $j$  e  $k$  codificados na fita. Não há criação dinâmica de estados nem duplicação de submáquinas em função do comprimento da palavra de entrada.

Assim, o número total de estados da máquina é fixo e determinado exclusivamente pela especificação do algoritmo.

Portanto, a complexidade da MT2 em número de estados é constante:

$$|Q| \in O(1)$$

## 3.5. Exemplos passo a passo

A seguir, apresentam-se exemplos ilustrativos do funcionamento da MT2, explicitando o conteúdo das fitas auxiliares, incluindo as representações unárias do total e da metade, bem como a aplicação correta do critério de maioria absoluta.

### 3.5.1. Exemplo 1

Considere a entrada:

$$A^3 B^2 C^1$$

Após a leitura da fita de entrada, a MT2 registra, em notação unária, as quantidades de veículos em cada avenida:

- fita A: AAA
- fita B: BB
- fita C: C

O número total de veículos é:

$$3 + 2 + 1 = 6$$

A fita do total contém exatamente seis símbolos 1:

- fita total: 111111

Em seguida, a MT2 calcula o floor da metade do total. Obtém-se:

$$\frac{6}{2} = 3$$

resultando na seguinte fita da metade:

- fita metade: 111

Na etapa de verificação de maioria absoluta, a MT2 compara, de forma não destrutiva, a fita da avenida A com a fita da metade, realizando uma varredura simultânea até o símbolo branco \_.

Ambas as fitas atingem o símbolo \_ ao mesmo tempo, pois possuem o mesmo comprimento. Assim, a quantidade de veículos da avenida A é igual à metade do total, e, portanto, **não caracteriza maioria absoluta**.

Não havendo maioria absoluta, a MT2 prossegue para a fase de comparações diretas entre as quantidades individuais. Como a avenida A possui estritamente mais veículos que as avenidas B e C, ela é selecionada como prioritária.

Ao final do processamento, o símbolo A é escrito na fita de saída, e a máquina alcança o estado final de aceitação.

### 3.5.2. Exemplo 2

Considere a entrada:

$$A^2B^2C^2$$

Após a leitura da entrada, as fitas auxiliares contêm:

- fita A: AA
- fita B: BB
- fita C: CC

O número total de veículos é:

$$2 + 2 + 2 = 6$$

logo, a fita do total é:

- fita total: 111111

A metade do total é:

$$\frac{6}{2} = 3$$

resultando em:

- fita metade: 111

Na verificação de maioria absoluta, nenhuma das fitas A, B ou C possui comprimento estritamente maior que o da fita da metade. Assim, nenhuma avenida apresenta maioria absoluta.

A MT2 então realiza comparações diretas entre as quantidades armazenadas nas fitas auxiliares. As comparações indicam empate entre as três avenidas.

Nesse caso, a decisão final segue o critério de desempate pré-definido pela máquina, baseado na ordem de prioridade:



$$A > B > C$$

Consequentemente, a avenida A é selecionada, o símbolo A é escrito na fita de saída, e o processamento é encerrado no estado final de aceitação.

### 3.6. Comparação com a MT1

A MT1, definida como uma Máquina de Turing determinística de fita única, realiza o processamento da entrada e todas as etapas de decisão utilizando apenas uma fita de trabalho. Como consequência, diversas informações intermediárias precisam ser acessadas repetidamente por meio de rebobinamentos completos da fita.

Em particular, na MT1:

- a contagem das ocorrências de cada símbolo exige múltiplas varreduras da fita de entrada;
- o cálculo do total e da metade do número de veículos envolve deslocamentos frequentes do cabeçote ao longo de uma fita de comprimento linear;
- as comparações entre quantidades são realizadas por meio de cancelamentos sucessivos e retornos ao início da fita, resultando em custos quadráticos no pior caso.

Essas operações implicam que símbolos da entrada e de estruturas auxiliares são revisitados um número linear de vezes, o que conduz a um tempo total de execução da ordem de:

$$\Theta(n^2)$$

Por outro lado, a MT2 explora o modelo de Máquina de Turing multifitas, dispondo de uma fita de entrada e três fitas auxiliares, o que permite separar fisicamente as informações relevantes e evitar rebobinamentos desnecessários.

Na MT2:

- a leitura da entrada é realizada em uma única varredura;
- as quantidades individuais são armazenadas em fitas distintas;
- todas as comparações são feitas por varreduras simultâneas, sem modificações destrutivas nem retornos repetidos ao início das fitas.

Como resultado, cada fita é percorrida apenas um número constante de vezes, e todas as etapas do processamento possuem custo linear no tamanho da entrada.

Assim, o tempo total de execução da MT2 é:

$$T(n) \in O(n)$$

Essa diferença evidencia o ganho assintótico proporcionado pelo uso do modelo multifitas: embora ambos os modelos sejam computacionalmente equivalentes em termos de linguagens reconhecíveis, a MT2 apresenta eficiência assintoticamente superior em relação à MT1 para o problema considerado.

## 4. Considerações finais

Nesse projeto, pudemos explorar as limitações e potencialidades das Máquinas de Turing como abstrações computacionais, da resolução de problemas lógico-matemáticos de natureza elementar, no percurso percebemos a robustez do módulo de Turing, à medida que fomos capazes de arquitetar soluções para questões como contagem e comparação de cardinalidade, além de emular estruturas típicas de linguagens de programação, como mecanismos de tomada de decisão e enlases de repetição.

Por esse caminho, também nos deparamos com as frustrações e vantagens inerentes a cada variante das MTs, não só no que tange a complexidade temporal, como também a clareza e concisão das soluções propostas.

Inicialmente, a implementação com uma única fita logo mostrou uma complicação aparante: a impossibilidade de estruturar diferentes operações em paralelo, forçando-nos a repetir diversas leituras, escritas e até etapas de higienização sobre a fita. Durante esse processo, optamos por uma abordagem modular, e tomamos decisões que não necessariamente eram as mais eficientes, mas que facilitavam a compreensão e a organização do código. Esse conjunto de circunstâncias culminou numa complexidade quadrática, o que reforçou a percepção de que a MT1, apesar de ser um modelo fundamental, pode não ser a mais adequada para todos os tipos de problemas.

Em seguida, o mesmo problema foi reespecificado no modelo de Máquina de Turing determinística de múltiplas fitas (MT2). A separação física das informações em uma fita de entrada e fitas auxiliares independentes possibilitou a execução das principais operações — leitura, contagem, cálculo da metade e comparações — por meio de varreduras lineares e não destrutivas. Como consequência, todas as etapas do processamento da MT2 foram demonstradas como executáveis em tempo linear, resultando em uma complexidade total de  $O(n)$ , sem a necessidade releituras ou limpezas.

A comparação entre MT1 e MT2 expõe de forma clara o ganho assintótico proporcionado pelo modelo multifitas. Embora ambos os modelos sejam equivalentes em termos de poder computacional e linguagens reconhecíveis, a MT2 apresenta eficiência significativamente superior para o problema proposto, reforçando a importância da escolha adequada do modelo de computação na análise de algoritmos.

Além da análise temporal, o trabalho também demonstrou que o número de estados da MT2 é constante e independente do tamanho da entrada, o que reforça a viabilidade estrutural da solução proposta. Os exemplos passo a passo apresentados complementam a descrição formal, tornando explícita a evolução do conteúdo das fitas e o processo de decisão em diferentes cenários de entrada.

Por fim, este estudo destaca como modelos clássicos da computação teórica podem ser aplicados à formalização de problemas do mundo real, como o controle inteligente de tráfego urbano. A abordagem adotada demonstra que, mesmo em sistemas abstratos como Máquinas de Turing, é possível capturar regras práticas de decisão, analisar corretude e eficiência, e comparar soluções sob diferentes arquiteturas computacionais,

contribuindo tanto para a formação teórica quanto para a compreensão de aplicações computacionais modernas.