

# **Controle de Tráfego Urbano Inteligente**

Trabalho Final do Curso de Teoria da Computação

Caio Mello - XXXX  
João Roberto - 222217111  
Jonas - ZZZZZZZZ  
Leandro - ZZZZZZZZ  
Vinicius - YYYYYYYYYY

,

**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.	Máquinas de Turing .....	6
2.1.	Máquina de uma única fita ( $MT_1$ ) .....	7
2.1.1.	Módulos e submódulos .....	7
2.1.2.	Módulo $\alpha$ .....	8
2.1.3.	Submódulo $\psi$ .....	9
2.2.	Máquina na íntegra .....	14
2.3.	Análise de complexidade .....	14
2.4.	Exemplos .....	14
3.	Máquina de múltiplas fitas .....	14
3.1.	Módulos .....	14
3.2.	Máquina na íntegra .....	14
3.3.	Análise de complexidade .....	14
3.4.	Exemplos .....	14

## 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.**

### 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  $n \in \mathbb{N}$

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

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

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

Será útil também a definição de uma função para cálculo da cardinalidade de uma letra, construída segundo o descrito abaixo:

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

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

$$\gamma_x(w) = |\{i \in \{1, \dots, |w|\} \wedge \varphi_i = x\}|$$

Onde  $\gamma$  é a função de cardinalidade.

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

A partir do enunciado da questão, abstraímos duas propriedades que nosso automóvel deve computar, as chamaremos de A e B.

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  $w$ :**

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

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

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 automômato.

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,

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

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

$$\gamma_x(w) > \gamma_y(w) + \gamma_z(w) \tag{\Delta}$$

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 automômato compute o [corolário  \$\Delta\$](#) :

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

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

## 2. Máquinas de Turing

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’.

Dada a definição de uma MT:

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

$E$ : Conjunto finito de estados

$\Sigma$ : Alfabeto de entrada

$\Gamma$ : Alfabeto da fita

$\delta$ : Funções de transição

$i$ : Estado inicial

$F$ : Conjunto de estados finais

Definimos um módulo  $\mathbb{M}$  de MT como:

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

Onde,

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

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

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

$e_{\text{out}}$ : 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  $e_{\text{out}} \subset F$ . 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.

## 2.1. Máquina de uma única fita ( $MT_1$ )

Nossa máquina de uma fita só, ou  $MT_1$ , é definida como:

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

Tal que,

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

$$\Gamma = \{x_y \mid \forall x, y \in \Sigma\} \cup \{\varepsilon, <\}, \text{ ou seja } \{a, b, c, a_a, a_b, a_c, b_a, b_b, b_c, c_a, c_b, c_c, <, \varepsilon\}$$

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

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

$\varepsilon$  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:

$$x \in \Sigma \mid F_x : \text{Sinaleira prioriza avenida } x$$

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

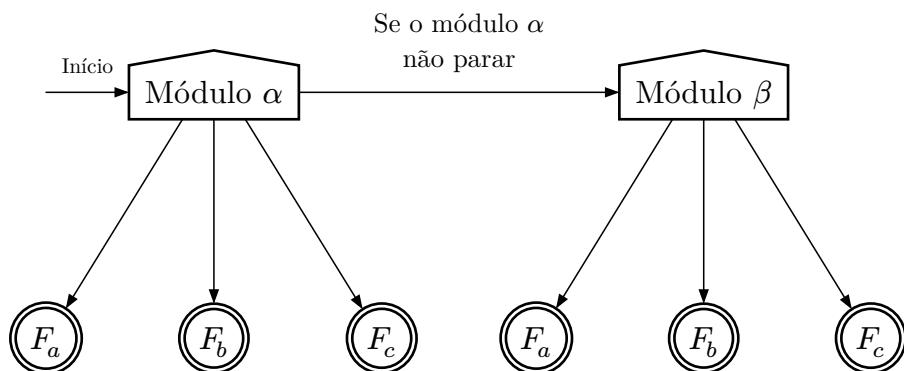
$MT_1$  tem dois módulos principais:  $\alpha$  e  $\beta$  os quais serão estabelecidos e destrinchados a seguir.

### 2.1.1. Módulos e submódulos

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 automônto 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 tem o formato:



Fluxograma 1: Arquitetura do automônto

### 2.1.2. 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 automômato 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](#).

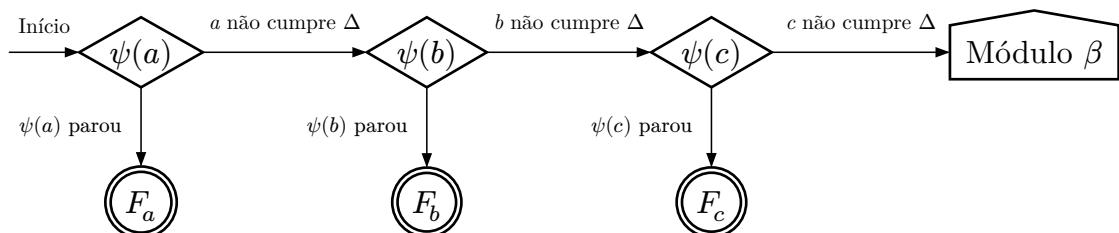
Portanto, efetuamos essa avaliação para as letra na ordem “a”, “b” e, então, “c”, implicitamente zelando pela ordem de desempate.

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

Quebramos cada avaliação de uma letra em um submódulo  $\psi$ , tal que:

$\psi(x)$  : Verifica se  $x$  cumpre a propriedade  $\Delta$ , caso positivo, o automômato pára em  $F_x$ .

Dando ao interior de  $\alpha$  o formato:



Fluxograma 2: Formato do Módulo  $\alpha$

### 2.1.3. Submódulo $\psi$

Como já definimos,  $\psi$  deve verificar se vale para algum  $x$ :

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

Para tal, devemos comparar a **cardinalidade** de  $x$  com a de  $y$  e  $z$ .

Com esse objetivo, nosso automato irá, efetivamente, **mapear** cada ocorrência de  $x$  com uma de  $y$  e de  $z$ .

A natureza dessa correspondência caracteriza diretamente a relação entre as cardinalidades envolvidas.

Considere  $X_w$  como o conjunto de aparições de  $x$  em  $w$ , e  $Y_w$  e  $Z_w$  como conjuntos análogos para demais letras. Formalizando:

$w$  é palavra,  $|w| = n$ , e  $w = (a_1, \dots, a_n)$  onde  $a_i \in \Sigma = \{a, b, c\}$

$$X_w = \{i \mid a_i = x\}, Y_w = \{i \mid a_i = y\}, Z_w = \{i \mid a_i = z\}$$

Teremos que a computação de  $\Delta$  é, essencialmente, a contagem dos conjuntos  $X_w$  e  $Y_w \cup Z_w$  – chamaremos essa união de  $\text{YZ}_w$

Com esse fim,  $\psi(x)$  buscará formar pares únicos de  $x$  com  $y$ 's e  $z$ 's, de maneira que não sejam repetidos os elementos envolvidos em cada par já formado. De maneira formal:

$$\psi(x) : X_w \rightarrow \text{YZ}_w$$

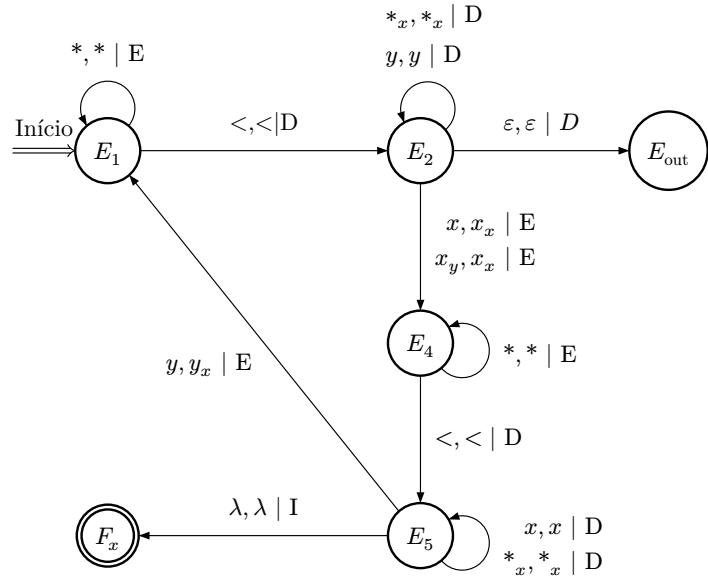
$$\forall i, j \in X_w, \psi(i) = \psi(j) \Rightarrow i = j$$

Assim, de acordo ao definido pela teoria dos Conjuntos, caso verifiquemos:

1. bijeção, intuimos que  $\gamma_x(w) = \gamma_y(w) + \gamma_z(w)$ ;
2. injeção, mas não sobrejeção,  $\gamma_x(w) < \gamma_y(w) + \gamma_z(w)$ ;
3. sobrejeção, mas não injeção,  $\gamma_x(w) > \gamma_y(w) + \gamma_z(w)$

Portanto, basta que nossa MT compute essa função de pareamento para que seja verificada a validade de  $\Delta$ .

Fixo que  $x, y \in \Sigma$  onde  $x \neq y$ , o submódulo  $\psi(x)$  terá o formato:



Submódulo 1: Função  $\psi$

Onde  $F_x$  simboliza a parada total da MT1 na decisão de priorizar a avenida  $x$ .

Enquanto  $E_{\text{out}}$  será:

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

Ou seja,  $\psi(x)$  deve

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

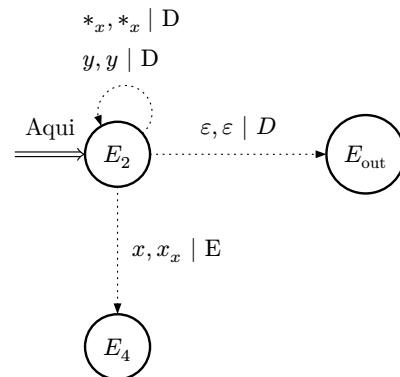
$$\text{Sair por } E_{\text{out}} \Leftrightarrow \gamma_x(w) \leq \gamma_y(w) + \gamma_z(w)$$

Vamos então destrinchar cada estado da máquina e suas transições associadas para verificar a computação das propriedades desejadas e futuramente computar a complexidade.

Passo 1 - Rebobina ao início da fita.

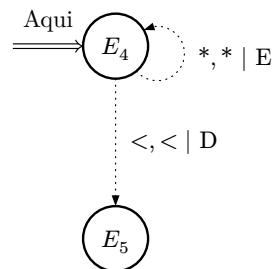


Passo 2 - Avança pela fita passando por todos  $y$  e  $x_x$  – que são os  $x$ 's já pareados – até encontrar um  $x$  ou o fim da fita.

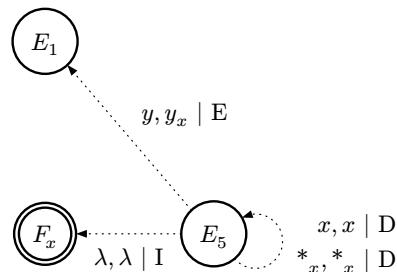


- a) Caso ache um  $x$  ou  $x_y$ , o sobreescrivemos por  $x_x$  e seguimos para o passo 3.
- b) Caso não ache um  $x$ , escapamos para  $E_{\text{out}}$ .

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, sabemos que  $x$  cumpre com a [a propriedade C](#), pára no estado final  $F_x$  e cessamos computação.
- b) Caso encontre, sobreescrivemos o  $y$  na fita com  $y_x$  e retornamos ao passo 1.

Um aspecto relevante é que esse módulo é um *loop* com duas interrupções possíveis: 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,

$$n, i \in N$$

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

$$w = k_0 * \dots * k_n$$

Os passos 2.a) e 4.b) marcam os elementos de forma  $x_x$  e  $y_x$ , respectivamente, construindo os conjuntos:

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

$$\bar{X}_w = \{j \mid \forall j \in N \text{ tal que } 0 \leq j \leq n \text{ e } k_j \neq x\}$$

Como é feita uma substituição de elemento por vez, os conjuntos são proceduralmente compostos em paralelo. Dessa forma, independente da iteração do loop, em 2.a),  $|X_w| = |\bar{X}_w| + 1$ . Até que em 4.b), é encontrado mais um  $y$  e  $|X_w| = |\bar{X}_w|$ .

Somente em uma das interrupções, 2.b) ou 4.a), ocorre um disruptura dessas igualdades.

Em 2.b), não encontramos mais  $x$  e escapamos para  $E_{out}$ , como interrompemos o processamento do módulo, não verificamos se existem mais  $y$ , mas temos certeza da cardinalidade de  $x$ . Obtendo  $|\bar{X}_w| = |X_w|$  e, portanto,  $\gamma(x)_w = \gamma(y)_w + \gamma(z)_w$ , entretanto, como não buscamos mais  $y$ , podemos afirmar de maneira mais forte que  $\gamma(x)_w \leq \gamma(y)_w + \gamma(z)_w$ .

Já na parada em 4.a), temos do passo 2.a) que  $|X_w| = |\bar{X}_w| + 1$  e interrompemos o processamento em  $F_x$ . Neste caso,  $\gamma(x)_w \geq \gamma(y)_w + \gamma(z)_w + 1$ , o que equivale a  $\gamma(x)_w > \gamma(y)_w + \gamma(z)_w$ . Neste caso, computamos com sucesso que vale a propriedade  $\Delta$  no momento da parada em  $F_x$ .



## **2.2. Máquina na íntegra**

ipsi literis ipsi literis ipsi literis ipsi literis ipsi literis ipsi literis

## **2.3. Análise de complexidade**

ipsi literis ipsi literis ipsi literis ipsi literis ipsi literis ipsi literis

## **2.4. Exemplos**

ipsi literis ipsi literis ipsi literis ipsi literis ipsi literis ipsi literis

# **3. Máquina de múltiplas fitas**

## **3.1. Módulos**

ipsi literis ipsi literis ipsi literis ipsi literis ipsi literis ipsi literis

## **3.2. Máquina na íntegra**

ipsi literis ipsi literis ipsi literis ipsi literis ipsi literis ipsi literis

## **3.3. Análise de complexidade**

ipsi literis ipsi literis ipsi literis ipsi literis ipsi literis ipsi literis

## **3.4. Exemplos**

ipsi literis ipsi literis ipsi literis ipsi literis ipsi literis ipsi literis