

Controle de Tráfego Urbano Inteligente

Trabalho Final do Curso de Teoria da Computação

João Roberto - 222217111

Caio Mello - XXXX

Vinicius - YYYYYYYYYY

Ciclano - ZZZZZZZZ

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	6
2.1. Módulos e submódulos	6
2.1.1. Dentro do Módulo α	6
2.1.2. Função ψ	7
2.2. Máquina na íntegra	7
2.3. Análise de complexidade	7
2.4. Exemplos	7
3. Máquina de múltiplas fitas	7
3.1. Módulos	7
3.2. Máquina na íntegra	7
3.3. Análise de complexidade	7
3.4. Exemplos	7

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.

[illegible]

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

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

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

Onde γ 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 automômato deve computar, as chamaremos de A e B.

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

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

$$x \in \{a, b, c\} \mid \gamma_x(w) > \frac{\gamma_a(w) + \gamma_b(w) + \gamma_c(w)}{2} \quad (\text{A})$$

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

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

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

$$\begin{aligned} x, y, z \in \{a, b, c\} \wedge y \neq z \wedge z \neq x \wedge x \neq z \\ \gamma_x(w) > \gamma_y(w) \wedge \gamma_x(w) > \gamma_z(w) \end{aligned} \quad (\text{B})$$

- 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 ocorrer 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,

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

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

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

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 automôto compute [propriedade C](#) para que também valha a [condição 1](#) do enunciado.

2. Máquina de uma única fita

2.1. Módulos e submódulos

Nossa máquina de uma única fita, a qual iremos tratar por **MT1**, terá dois módulos principais: α e β .

O módulo α será responsável por verificar se a [condição 1](#) do enunciado para alguma letra a, b ou c através da [propriedade C](#).

Cada estado final da máquina representará a decisão de qual único sinal que deve ter prioridade dado um estado como *input*.

Caso nenhum estado final seja alcançado dentro de A, o automômato recorrerá ao módulo B, onde avaliará a [condição 2](#) do problema.

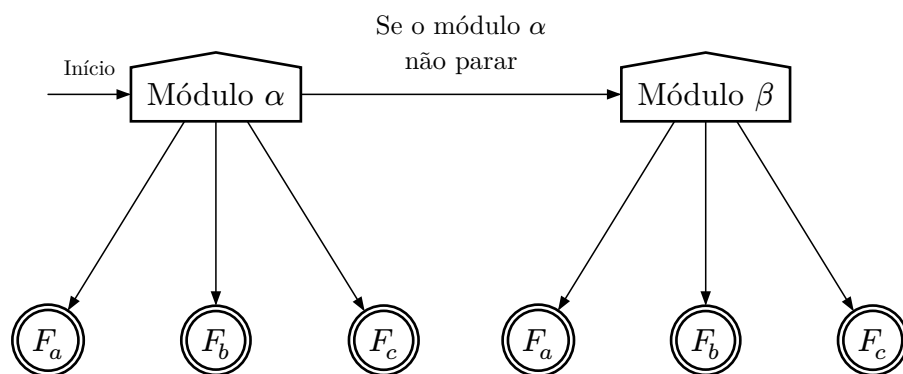


Diagrama 1: Arquitetura do automômato

2.1.1. Dentro do Módulo α

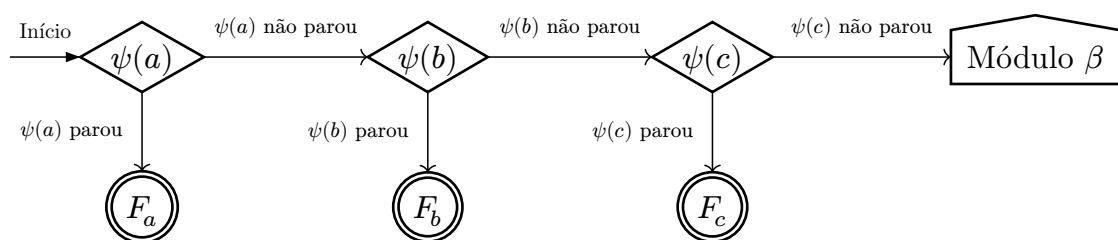
O módulo α irá analisar se [a propriedade C](#) vale para alguma letra em $\{a, b, c\}$. Essa operação será feita linearmente, um símbolo por vez até que paremos em um estado final ou caso os símbolos a serem lidos se esgotem.

Neste último caso entendemos que ninguém cumpriu com as condições necessárias para parada e recorreremos ao módulo β .

Dessa forma, abstraímos um submódulo ψ tal que:

$\psi(x)$: Computa se x cumpre a propriedade C, caso positivo, o automômato pára.

Verificaremos na ordem $\psi(a)$, $\psi(b)$, $\psi(c)$. Garantindo a condição de desempate definida em [I](#). Visto que caso alguma avaliação de ψ pare, as demais não serão computadas. Assim preservamos a ordem de prioridade nos empates.



Fluxograma 1: Formato do Módulo A

2.1.2. Função ψ

Essa parte da máquina deve verificar se vale:

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

Tal que,

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

Para isto, definimos dois conjuntos X e \overline{X} da seguinte forma:

Sejam,

$$n \in \mathbb{N} \mid x, \gamma_n \in \{a, b, c\} \mid w \text{ é uma palavra.} \mid w = \gamma_0 * \dots * \gamma_n$$

Então temos,

$$X = \{\forall \gamma_i \in w \mid \gamma_i = x\}, \text{ e}$$

$$\overline{X} = \{\forall \gamma_i \in w \mid \gamma_i \neq x\}$$

Basta que verifiquemos:

$$|X| > |\overline{X}|$$

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

2.2. Máquina na íntegra

ipsi literis 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 ipsi literis

2.4. Exemplos

ipsi literis 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 ipsi literis

3.2. Máquina na íntegra

ipsi literis 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 ipsi literis

3.4. Exemplos

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