Universidade Federal Do Maranhão - UFMA

Curso de Engenharia da Computação-ECP

Máquina de Turing: Multiplicadora de Binários

São Luís-MA

2025

Máquina de Turing: Multiplicadores de Binários

Keven Gustavo Dos Santos Gomes

Lucas Emanoel Amaral Gomes

RESUMO:

Este estudo apresenta a execução de uma Máquina de Turing (MT) que realiza a multiplicação de dois números binários, a MT foi implementada com o intuito de reproduzir o processo clássico de multiplicação por meio de operações de deslocamento e adição, embora adaptadas para o sistema binário. A máquina funciona sobre uma fita infinita onde os números binários de entrada são armazenados inicialmente, separados por um símbolo especial. Ao longo da execução, a MT passa pelas seguintes etapas: Leitura e tratamento dos dados, Multiplicação e Armazenamento do resultado. A implementação foi testada para diversos pares de números binários, mostrando a corretude e a eficiência do algoritmo. O trabalho também aborda as limitações práticas da MT, como tempo para execução e o espaço utilizado na fita, em comparação a métodos computacionais tradicionais. Conclui-se que, ainda que a MT seja um modelo teórico, é viável sua aplicação para realização de operações aritméticas como a multiplicação binária, ilustrando claramente os fundamentos da computação.

Palavras-chave: Máquina de Turing, Multiplicação binária, Números binários, Fita infinita, Operações de deslocamento, Adição binária

1 INTRODUÇÃO

A Máquina de Turing (MT), idealizada por Alan Turing em 1936, está dentre os fundamentos da teoria da computação, servindo como um modelo teórico para se entender os limites e capacidades dos algoritmos e dos sistemas computacionais, sua simplicidade conceitual e seu poder de simular qualquer procedimento algorítmico a tornaram uma ferramenta fundamental no estudo dos problemas computacionais, desde os mais simples até os mais complexos.

Este trabalho aborda a aplicação da Máquina de Turing para realizar uma operação aritmética fundamental, a multiplicação de números binários é uma operação central na computação, estando presente em várias aplicações práticas, como no processamento de dados, na criptografia e na implementação de circuitos lógicos, apesar de ser um processo bem definido e bastante utilizado, sua implementação em uma Máquina de Turing oferece desafios interessantes, uma vez que a MT opera de maneira sequencial sobre uma fita infinita, sem a capacidade de acessar a memória de maneira aleatória ou realizar operações de maneira paralela, isso traz a necessidade de criação de um algoritmo que simule, passo a passo, o processo de multiplicação, utilizando tão somente operações básicas como leitura, escrita e movimentação na fita.

O algoritmo desenvolvido simula o método tradicional de multiplicação, baseado em deslocamentos e somas, adaptado de forma a atuar no sistema binário, a MT é projetada para processar os números de entrada, gravados na fita, e para realizar as operações necessárias para a produção do produto final, além disso, discutimos as limitações práticas desse procedimento, tais como tempo de execução e uso de espaço na fita, ao se fazer um paralelo com métodos computacionais tradicionais.

2 METODOLOGIA

A elaboração do presente código, que implementa a multiplicação de dois números binários por meio de uma Máquina de Turing, seguiu uma abordagem de pesquisa aplicada com base em estudos de computabilidade e teoria da computação, fundamentou-se, principalmente, nos conceitos propostos por Turing (1936), no que se refere à definição de uma máquina teórica capaz de ler símbolos em uma fita, escrever novos símbolos e transitar entre estados de forma sistemática., a seguir, descrevem-se os procedimentos metodológicos adotados:

2.1 Caracterização da Pesquisa

O estudo caracteriza-se como aplicado, pois objetiva a construção de uma solução computacional específica (a multiplicação binária), fundamentada em princípios teóricos de Máquinas de Turing, optou-se por uma pesquisa experimental, uma vez que se desenvolveu um protótipo em linguagem Python para validar a funcionalidade do modelo, além disso, empregou-se uma abordagem qualitativa na análise do funcionamento e no refinamento iterativo do código.

2.2 Fundamentação sobre Máquina de Turing

Para embasar o desenvolvimento, foram revisados os conceitos de autômatos e linguagens formais, com ênfase nas Máquinas de Turing, que neste caso é uma determinística, ou seja, possui no máximo uma entrada para cada combinação de símbolo e estado, e tem uma fita única infinita, então para a sua implementação em python, adotou-se a estratégia de representar cada transição da Máquina de Turing por meio de uma tupla contendo cinco elementos:

1. Estado atual;
2. Símbolo lido;
3. Novo símbolo a ser escrito;
4. Direção de movimentação da cabeça de leitura/escrita (esquerda, direita ou sem movimento);
5. Novo estado.

A partir desses elementos, construiu-se um dicionário (self.transitions) para mapear cada par (EstadoAtual,Símbolo Lido) em um conjunto de ações(Novo Símbolo, Direção, Estado Novo), essa modelagem é inspirada na definição formal de Máquina de Turing, segundo a qual cada transição é uma quíntupla que determina o comportamento da máquina.

2.3 Fundamentação da Multiplicação de Binários

A multiplicação de números binários baseia-se nos mesmos princípios aritméticos empregados na multiplicação decimal, diferenciando-se apenas pela base utilizada, em vez de se operar com dez dígitos (0 a 9), trabalha apenas com dois (0 e 1), o que implica um conjunto simplificado de regras de soma e vai-um (carry), na prática, cada bit do segundo número determina se o primeiro número será somado à acumulação parcial ou não:

- **Bit 0:** Se o bit atual do segundo número for 0, significa que não há contribuição do primeiro número na posição binária correspondente, dispensando-se a soma.
- **Bit 1:** Se o bit atual for 1, então o primeiro número deve ser adicionado ao resultado parcial, representando a contribuição do respectivo dígito.

Adicionalmente, a cada passo de análise de um bit do segundo número, o primeiro número é "deslocado" para a esquerda, equivalente a multiplicá-lo por 2, este deslocamento simula o comportamento que, na multiplicação decimal, seria mover uma casa para a esquerda ao passar para o próximo algarismo, a soma final de todos os deslocamentos apropriados gera o produto binário completo.

2.4 Estrutura e Planejamento do Código

O desenvolvimento do código foi planejado para separando parte lógica (transições) da parte estrutural (classe que executa as transições):

- **Definição das Transições**: Foi criado um bloco textual (tm_logic) descrevendo cada regra de comportamento da Máquina de Turing, no formato <estado_atual> <símbolo_atual> <novo_símbolo> <direção> <novo_estado>, essa formatação permite que cada linha seja lida e armazenada em um dicionário de transições, viabilizando a escalabilidade e a clareza do código.
- **Classe TuringMachine**: tem o objetivo de gerenciar a execução da máquina, possuindo métodos para: analisar e armazenar as regras de transição, a partir de uma definição textual já fundamentada anteriormente, executar a simulação da Máquina de Turing, manipulando a fita conforme as transições definidas, como manter o estado atual,

controlar a fita e a posição da cabeça de leitura/escrita, e o último método que exibe os passos da execução, permitindo a visualização detalhada do processamento de cada etapa.

A entrada da máquina consiste em uma string contendo dois números binários separados por 'S', que é então processada pela classe TuringMachine seguindo o processo definido pelas transições, até o estado final definido onde então a saída é o produto desses números representado em binário na fita final, que foi obtido por este processo.

2.5 Procedimentos de Implementação

1. **Leitura das transições**: Foi desenvolvido o método **parse_logic(logic)**, responsável por ler cada linha de uma string contendo a lógica da máquina (os estados e transições) e inserir as informações no dicionário de transições.
2. **Execução passo a passo**: O método **run(input_tape, show_steps=False)** recebe a fita de entrada, converte-a em uma lista de caracteres e inicia a simulação a partir de um estado inicial ('0').
3. **Processamento de cada transição**: A cada iteração, o código verifica se há uma transição correspondente ao par(**Estado Atual**, **Símbolo Atual**), se encontrada, executa a escrita do novo símbolo na fita, move a cabeça de acordo com a direção especificada e altera o estado para o novo estado definido.
4. **Controle de terminação**: A máquina encerra a execução (estado "**halt**") ao encontrar um estado que começa com a palavra "**halt**", ou caso se ultrapasse um número limite de iterações (1000 passos), prevenindo loops infinitos.
5. **Limpeza da fita**: Ao final, símbolos em branco (**'_'**) são removidos das extremidades da fita, retornando somente o resultado do processamento, neste caso, o produto binário.

2.6 Lógica de Transição da Máquina

A lógica de transição da Máquina de Turing define, para cada par (Estado Atual, Símbolo Atual), qual símbolo deve ser escrito na fita, para onde a cabeça de leitura deve se mover (esquerda, direita ou permanecer parada) e qual será o próximo estado, abaixo descreve-se as principais rotinas de transição utilizadas para efetuar a multiplicação binária:

1. **Inserção do símbolo + (estados 0 e init)**
    - A máquina inicia no estado 0, lendo o primeiro bit do primeiro número binário. Em seguida, move a cabeça para a esquerda, insere o símbolo + no início da fita (estado init) e então avança para a direita.
    - **Propósito**: criar uma área de contagem ou somatório parcial no lado esquerdo da fita, onde o resultado será acumulado.
2. **Leitura do segundo número (estado right)**
    - No estado right, a máquina percorre a fita para a direita até encontrar um espaço em branco (_).
    - **Propósito**: localizar o último bit do segundo número (por exemplo, no caso de 10S11, esse bit seria o 1 mais à direita) e preparar-se para processá-lo.
3. **Identificação do bit atual do segundo número (estado readB)**
    - Ao encontrar o bit final do segundo número, a máquina lê se ele é 0 ou 1.
        - Se for 0, não adiciona o primeiro número ao acumulador, apenas dobra o primeiro número (rotina doubleL).
        - Se for 1, chama-se sub-rotina addA, que fará a soma do primeiro número ao valor já acumulado.
    - **Propósito**: decidir se aquele bit contribui ou não para o resultado final, pois em binário a multiplicação equivale a somas condicionadas pelos bits 1.
4. **Rotina de soma (estado addA e subsequentes)**
    - Caso o bit do segundo número seja 1, a máquina passa para o estado addA, que desloca a cabeça pela região do primeiro número até encontrar o separador S.
    - Ao encontrar “S”, entra no estado read (e estados correlatos, como have0, have1, add0, add1, carry), responsáveis pela **soma binária** do primeiro número ao “tally” localizado à esquerda do +.
    - **Propósito**: implementar a soma bit a bit, incluindo o tratamento de vai-um (carry).
5. **Dobrando o primeiro número (estados doubleL e double)**
    - Após processar um bit do segundo número (seja 0 ou 1), a máquina **dobra** o primeiro número, adicionando um 0 ao seu final (equivalente a deslocar para a esquerda em binário).

- Essa rotina faz a varredura do primeiro número até o símbolo separador, insere 0 e então prossegue para organizar a fita (estado shift).
- **Propósito**: cada vez que se avança para o próximo bit do segundo número, o primeiro número é multiplicado por 2 para corresponder ao deslocamento binário.

6. **Deslocando o segundo número(estado shift)**
   - A máquina move o segundo número para a direita em uma célula, "descartando" o bit que acabou de ser processado.
   - Quando o segundo número se esgota, a transição é para o estado tidy, que finaliza a limpeza da fita.
   - **Propósito**: remover o bit processado e atualizar a fita para ler o próximo bit do segundo número (se existir).
7. **Limpeza e término (estado tidy e halt-done)**
   - Em tidy, a máquina apaga símbolos intermediários e, ao encontrar +, converte-o em _ para sinalizar o fim do cálculo.
   - Entra em halt-done, encerrando a execução.
   - **Propósito**: garantir que a fita final apresente apenas o resultado binário, sem resíduos de símbolos temporários.

Assim, cada parte da lógica de transição é responsável por um estágio da multiplicação binária: identificar bits do segundo número, somar o primeiro número ao acumulador quando o bit for 1, dobrar o primeiro número a cada iteração, deslocar o segundo número para o próximo bit e, finalmente, limpar a fita antes de parar. A soma repetida e o deslocamento binário são suficientes para efetuar a multiplicação, conforme previsto pela teoria de Máquinas de Turing.

2.7 Testes e Validação

Para verificar a correção do código, foram executados testes de multiplicação binária com entradas diversas, aumentado o número de bits dos números iniciais, o que então eleva a complexidade da operação, por exemplo:

- **10S11** (2 * 3 = 6)
- **101S111** (5 * 7 = 35)
- **1101S1100** (13 * 12 = 156)

Após cada execução, conferiu-se o resultado na fita final, comparando-o ao produto binário esperado. Essa validação segue o método de teste de caixa-preta, focado na verificação de resultados para entradas conhecidas, sem necessariamente inspecionar a implementação interna linha a linha.

3 RESULTADOS E DISCUSSÃO

Nesta seção, apresenta-se a análise dos resultados obtidos pelo simulador de Máquina de Turing para três casos de teste representativos, em cada teste foram fornecidos dois números binários separados por S, e o objetivo era verificar se a saída final na fita, após o processamento, correspondia ao produto binário correto, os experimentos foram conduzidos no ambiente Python, exibindo-se o número de passos e a fita resultante ao final de cada execução.

3.1 Caso 1: 10S11

O primeiro teste consistiu em multiplicar 10 (2 em decimal) por 11 (3 em decimal). Ao inserir a fita inicial 10S11_ no simulador, a máquina:

- Inseriu o símbolo + à esquerda para controle do somatório parcial.
- Percorreu a fita até o último bit do segundo número (1), detectando que era necessário adicionar o primeiro número ao acumulador.
- Realizou a rotina de soma (addA) para incluir o valor do primeiro número no total.
- Dobrou (multiplicou por 2) o primeiro número conforme cada bit processado, até que o segundo número se esgotasse.

Ao final, o resultado obtido foi 110, que corresponde a 6 em decimal, desta forma verificou-se a conformidade com a multiplicação esperada ($2 \times 3 = 6$), este caso demonstrou o correto funcionamento da máquina ao lidar com múltiplas iterações de soma e deslocamento (shift) mesmo em um exemplo relativamente simples.

3.2 Caso 2: 101S111

No segundo teste, avaliou-se a multiplicação de 101 (5 em decimal) por 111 (7 em decimal). A fita de entrada foi 101S111_, observou-se que:

- Para cada bit do segundo número (iniciando pelo menos significativo), a máquina identifica se o bit é 0 ou 1.
- Quando o bit era 1, a lógica de **addA** somou o primeiro número (no estado atual de duplicação) ao acumulador.
- Ao final de cada bit processado, o primeiro número era dobrado pela rotina double e reposicionado na fita pela sub-rotina shift.

Concluído o processo, a fita resultante foi 100011, que em decimal corresponde a 35, a comparação com o resultado obtido por métodos tradicionais (por exemplo, a multiplicação direta de 5 × 7) confirmou a exatidão da simulação.

3.3 Caso 3: 1101S1100

Por fim, empregou-se o teste 1101S1100_, que representa a multiplicação de 1101 (13 em decimal) por 1100 (12 em decimal). Esse caso é mais extenso e visa avaliar se a máquina mantém o comportamento correto ao lidar com um número maior de bits:

1. A máquina percorreu a fita para identificar o bit atual do segundo número (1100).
2. Cada vez que encontrava 1, somava o primeiro número (1101, depois duplicado para 11010, 110100 etc.) ao acumulador, atualizando o valor parcial.
3. Quando o bit era 0, simplesmente realizava a duplicação do primeiro número, sem adicionar ao acumulador.
4. Ao final, removeu-se todo o conteúdo temporário e chegou-se ao resultado final em binário.

O resultado obtido foi 10011100, que em decimal é 156 (13 × 12). Esse valor coincide com o esperado, validando o algoritmo para números binários com maior quantidade de dígitos.

3.4 Discussão dos Resultados

A partir dos três casos, verificou-se que a máquina de Turing implementada executou corretamente a lógica de multiplicação binária. Em cada cenário:

- **Rotina de Soma**: Foi fundamental para acumular o valor do primeiro número toda vez que um bit 1 era detectado no segundo número.
- **Rotina de Duplicação (dobrar)**: Garantiu que o primeiro número fosse deslocado em valor conforme se avançava na leitura dos bits do segundo número.
- **Deslocamento (shift)**: Ajustou o segundo número para descartar o bit já processado e prepará-lo para a próxima iteração.

Observou-se que o tempo de execução (ou quantidade de passos) cresce proporcionalmente ao produto dos tamanhos dos dois números, o que é coerente com a abordagem de "soma repetida", não foram identificados comportamentos inesperados, e a limpeza final da fita assegurou a exibição somente do produto binário.

Estes resultados reforçam a tese de que, ainda que simples, a Máquina de Turing proposta é capaz de ilustrar a computação de uma operação aritmética fundamental, confirmando tanto a validade de sua modelagem quanto o alinhamento com os princípios formais de autômatos e linguagens.

4 CONSIDERAÇÕES FINAIS

Através deste projeto foi demonstrado que a Máquina de Turing (MT), é apropriada para executar a multiplicação de dígitos binários, reforçando a versatilidade e a força desse modelo teórico para simular processos algorítmicos complexos, a implementação foi feita em linha com o mesmo raciocínio do algoritmo de multiplicação ordinário, que foi adaptado para o sistema binário, utilizando apenas as operações elementares da MT, como ler, escrever e mover a fita, os resultados obtidos confirmaram a corretude do algoritmo, que realizou a multiplicação de pares de números binários com precisão.

Também foi observado características importantes sobre a MT, entre elas a flexibilidade, pois a implementação permite combinar sub-rotinas para executar lógicas complexas, a próxima significativa é a universalidade, já que qualquer função computável pode ser expressa e processada por uma Máquina de Turing, por fim, a confiabilidade também se sobrepõe, porque o código que desenvolvemos possui mecanismos de segurança que garantem o funcionamento correto da operação, assim o projeto corrobora os princípios fundamentais desse modelo teórico de computação.

REFERÊNCIAS

Vista do Multiplicando números binários com Máquinas de Turing: Interdisciplinaridade no Ensino de Computação. Disponível em:<https://sol.sbc.org.br/index.php/sbie/article/view/22503/22327 >. Acesso em: 11 fev. 2025.

MCAMILAMP. GitHub - mcamilamp/Maquina-Turing. Disponível em:<https://github.com/mcamilamp/Maquina-Turing >. Acesso em: 11 fev. 2025.

Turing machine to Multiply two binary numbers – T4Tutorials.com. Disponível em:<https://t4tutorials.com/turing-machine-to-multiply-two-binary-numbers/>. Acesso em: 11 fev. 2025.

SIPSER, Michael. Introdução à Teoria da Computação. 3. ed. São Paulo: Cengage Learning, 2018.