Atividade Prática

Teoria da Computação e Compiladores

Luis Guilherme Busaglo Lopes
Matrícula: 20220086977

20 de outubro de 2025

Resumo

Este relatório apresenta a implementação e análise prática de diversos modelos de autômatos e máquinas de estados, desenvolvidos no simulador JFLAP. O objetivo é consolidar o conhecimento teórico sobre Autômatos Finitos Determinísticos (AFD), Autômatos Finitos Não-Determinísticos (AFND), Máquinas de Turing, Máquinas de Mealy e Moore, bem como Gramáticas Livres de Contexto (GLC). Cada modelo foi implementado seguindo as especificações teóricas e testado com diferentes entradas para validar seu funcionamento.

Conteúdo

1 Introdução

A Teoria da Computação é a área fundamental da Ciência da Computação que estuda os limites do que pode ser computado e como fazê-lo de forma eficiente. Nesta atividade prática, desenvolvemos e testamos diversos modelos computacionais essenciais:

- **Autômatos Finitos Determinísticos (AFD)**: Máquinas que leem entrada caractere por caractere e decide aceitar ou rejeitar baseado em uma sequência de transições determinísticas.
- **Autômatos Finitos Não-Determinísticos (AFND)**: Variação do AFD que permite múltiplas transições possíveis para o mesmo símbolo de entrada.
- **Máquinas de Turing**: Modelo computacional teórico que simula qualquer computador moderno, com fita infinita e cabeçote móvel.
- **Máquinas de Mealy e Moore**: Máquinas de estados com saída que geram resultados durante as transições ou estados.
- **Gramáticas Livres de Contexto (GLC)**: Formalismos para descrever linguagens através de produções gramaticais.

O simulador JFLAP foi utilizado para implementar, visualizar e testar cada um destes modelos.

2 Fundamentação Teórica

2.1 Autômatos Finitos Determinísticos (AFD)

Um Autômato Finito Determinístico é definido formalmente como uma quíntupla:

$$M = (Q, \Sigma, \delta, q_0, F)$$

Onde:

- Q é um conjunto finito de estados;
- Σ é um alfabeto (conjunto de símbolos de entrada);
- $\delta : Q \times \Sigma \to Q$ é a função de transição determinística;
- $q_0 \in Q$ é o estado inicial;
- $F \subseteq Q$ é o conjunto de estados finais (aceitação).

Um AFD processa uma cadeia de entrada símbolo por símbolo, seguindo transições determinísticas. A cadeia é aceita se, após ler todos os símbolos, o autômato termina em um estado final.

Exemplo Prático: O AFD implementado reconhece sequências de tokens de controle de acesso (ENTRA, SAI, ACESSO123, etc.), validando o fluxo de autenticação.

2.2 Autômatos Finitos Não-Determinísticos (AFND)

Um Autômato Finito Não-Determinístico estende o conceito de AFD permitindo:

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$$

Isto é, para um mesmo símbolo, pode haver múltiplas transições possíveis, e também permite transições vazias (ε-transições) que não consomem entrada.

Propriedade Importante: Todo AFND pode ser convertido para um AFD equivalente através da construção de subconjuntos.

Exemplo Prático: O AFND reconhece variações de palavras-chave como ENTRA, ENTRAR, ENTRO, utilizando a não-determinismo para explorar diferentes caminhos.

2.3 Conversão AFND para AFD

O processo de conversão utiliza a técnica de construção de subconjuntos:

1. Cada estado do novo AFD representa um subconjunto de estados do AFND;
2. O estado inicial é $\{q_0\}$ mais o fecho-;
3. Para cada estado e símbolo, calcula-se o conjunto de todos os estados alcançáveis;
4. Um estado é final se contém pelo menos um estado final do AFND original.

Esta conversão é importante pois AFDs são mais eficientes computacionalmente para reconhecimento de linguagens.

2.4 Máquinas de Mealy

Uma Máquina de Mealy é um autômato que produz saída durante as transições:

$$M = (Q, \Sigma, \Lambda, \delta, \lambda, q_0)$$

Onde:

- Q é o conjunto de estados;
- Σ é o alfabeto de entrada;
- Λ é o alfabeto de saída;
- $\delta : Q \times \Sigma \to Q$ é a função de transição;
- $\lambda : Q \times \Sigma \to \Lambda$ é a função de saída.

A saída depende tanto do estado quanto do símbolo de entrada.

Exemplo Prático: A máquina de Mealy implementada produz mensagens de status $(\text{EM}_ESPERA, ACESSO_CONCEDIDO, ACESSO_NEGADO) baseadonasequênciadeentradas.$

2.5 Máquinas de Moore

Uma Máquina de Moore é uma variação onde a saída depende apenas do estado:

$$M = (Q, \Sigma, \Lambda, \delta, \lambda, q_0)$$

Com $\lambda : Q \to \Lambda$ (saída associada ao estado, não à transição).

A diferença fundamental é que em Moore, a saída é conhecida ao entrar em um estado, enquanto em Mealy, a saída é gerada durante a transição.

Exemplo Prático: A máquina de Moore modela um sistema de controle de acesso onde cada estado representa uma condição específica (PROCESSANDO, CONCEDIDO, NEGADO).

2.6 Máquinas de Turing

Uma Máquina de Turing é definida como:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$$

Onde:

- Q é um conjunto finito de estados;
- Σ é o alfabeto de entrada;
- Γ é o alfabeto da fita (inclui símbolo em branco);
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ é a função de transição;
- q_0 é o estado inicial;
- q_{accept} é o estado de aceitação;

- q_{reject} é o estado de rejeição.

A Máquina de Turing possui uma fita infinita e um cabeçote que pode se mover para esquerda ou direita, sendo capaz de computar qualquer função computável.

Exemplo Prático: A máquina de Turing implementada valida códigos de acesso formato "ENTRAACESSO123"ou "ENTRAACESSO456", demonstrando processamento mais complexo que autômatos finitos.

2.7 Gramáticas Livres de Contexto (GLC)

Uma Gramática Livre de Contexto é um formalismos de geração de linguagens definido como:

$$G = (V, \Sigma, P, S)$$

Onde:

- V é um conjunto de variáveis (não-terminais);
- Σ é um alfabeto de terminais;
- P é um conjunto de produções da forma $A \to \alpha$;
- S é o símbolo inicial.

GLCs são mais expressivas que linguagens regulares, permitindo estruturas recursivas e aninhadas.

Exemplo Prático: A GLC implementada descreve o padrão de autenticação: primeiro ENTRA, seguido de um código de acesso (ACESSO123 ou ACESSO456).

3 Modelos Implementados

3.1 1. Autômato Finito Determinístico (AFD) - Controle de Acesso

3.1.1 Especificação

O AFD implementado reconhece sequências de tokens de controle de acesso:

- **Estados:** q_0 (inicial), q_1 (após ENTRA), q_2 (acesso concedido), q_3 (rejeitado)
- **Alfabeto:** Tokens como ENTRA, SAI, ACESSO123, ACESSO456
- **Linguagem:** Sequências válidas de autenticação

3.1.2 Regras de Aceitação

- **Aceitas:** ENTRA, SAI, ACESSO456
- **Rejeitadas:** ENTRA, SAI (transição inválida)

O autômato valida que o fluxo segue a ordem correta: entrada → código de acesso → saída.

3.2 2. Autômato Finito Não-Determinístico (AFND)

3.2.1 Especificação

O AFND reconhece variações de palavras-chave por caractere:

- **Linguagem:** $\{ENTRA, ENTRAR, ENTRO\}$
- **Permite:** Múltiplos caminhos de transição
- **Inclui:** ε-transições para flexibilidade

3.2.2 Exemplos de Processamento

- **Aceitas:** ENTRA, ENTRAR, ENTRO
- **Rejeitadas:** ENTRAO, ENTRAR0 (caracteres não esperados)

3.3 3. Conversão AFND para AFD

3.3.1 Processo

A conversão do AFND para AFD utiliza a técnica de construção de subconjuntos:

1. Fecho-ε do estado inicial;
2. Para cada símbolo, computar transições de todos os estados alcançáveis;
3. Marcar estados contendo estados finais como finais.

3.3.2 Resultado

O AFD resultante é equivalente ao AFND, mas determinístico e potencialmente mais otimizado para execução.

3.4 4. Máquina de Turing - Validador de Código

3.4.1 Especificação

A Máquina de Turing valida códigos no formato: PREFIX + CÓDIGO

- **Aceitas:** ENTRAACESSO123, ENTRAACESSO456
- **Rejeitadas:** ENTRAACESSO124, ENTRAACESSO789

3.4.2 Funcionamento

1. Lê a fita e valida o prefixo ENTRA;
2. Valida o código (123 ou 456);
3. Move o cabeçote conforme necessário;
4. Aceita ou rejeita baseado no resultado.

3.5 5. Máquina de Mealy - Gerador de Status

3.5.1 Especificação

A máquina de Mealy produz mensagens de status durante transições:

- **Estados:** Espera, Processando, Concedido, Negado
- **Entradas:** ENTRA, ACESSO123/456, SAI
- **Saídas:** EM$_E SPERA$, $ACESSO_C ONCEDIDO$, $ACESSO_N EGADO$

3.5.2 Exemplo de Execução

- Entrada: ENTRA → Saída: EM$_E SPERA$
- Entrada: ACESSO123 → Saída: ACESSO$_C ONCEDIDO$
- Entrada: SAI → Saída: ACESSO$_N EGADO$

3.6 6. Máquina de Moore - Controle de Estados

3.6.1 Especificação

A máquina de Moore associa saídas aos estados:

- q_0: EM$_E SPERA$
- q_1: PROCESSANDO
- q_2: CONCEDIDO
- q_3: NEGADO

A saída é definida pelo estado, não pela transição.

3.6.2 Diferença de Mealy

Moore é mais simples para implementação pois a saída é determinística por estado, enquanto Mealy permite saídas mais específicas por transição.

3.7 7. Gramática Livre de Contexto

3.7.1 Produções

```
S -> ENTRA CODIGO
CODIGO -> ACESSO123 | ACESSO456
```

3.7.2 Cadeias Geradas

- ENTRA ACESSO123
- ENTRA ACESSO456

Esta gramática é mais expressiva que um autômato finito pois permite estrutura hierárquica e composição de componentes.

4 Procedimento de Teste

4.1 Ambiente de Execução

Todos os modelos foram testados no simulador **JFLAP 7.1**, que permite:

- Simulação passo a passo de autômatos;
- Visualização de transições;
- Teste com múltiplas entradas;
- Conversão entre modelos (AFND → AFD).

4.2 Protocolo de Teste

Para cada modelo, executamos:

1. **Validação do Diagrama:** Verificar se o diagrama está correto e bem definido;
2. **Teste de Aceitação:** Inserir cadeias esperadas para serem aceitas;
3. **Teste de Rejeição:** Inserir cadeias esperadas para serem rejeitadas;
4. **Análise de Transições:** Examinar o caminho de execução passo a passo;
5. **Captura de Evidências:** Registrar screenshots dos resultados.

4.3 Critérios de Sucesso

Um modelo é considerado corretamente implementado se:

- Aceita todas as cadeias da linguagem esperada;
- Rejeita cadeias fora da linguagem;
- Não apresenta transições indefinidas para entradas válidas;
- O diagrama está coerente com a especificação teórica.

5 Resultados e Análise

5.1 Validação Prática

Cada modelo foi testado e validado conforme segue:

5.1.1 AFD - Controle de Acesso

- **Resultado:** Funciona corretamente
- **Aceita:** Sequências válidas em ordem
- **Rejeita:** Transições fora do padrão esperado
- **Observação:** Diagrama bem estruturado com estados claramente identificados

5.1.2 AFND com Variações

- **Resultado:** Funciona corretamente
- **Aceita:** ENTRA, ENTRAR, ENTRO
- **Rejeita:** Cadeias com caracteres inválidos
- **Observação:** O não-determinismo é explorado corretamente pelo simulador

5.1.3 Conversão AFND → AFD

- **Resultado:** Equivalência preservada
- **Observação:** O AFD resultante reconhece exatamente a mesma linguagem do AFND
- **Nota:** Demonstra a importância teórica de que toda linguagem aceita por AFND também é aceita por AFD

5.1.4 Máquina de Turing

- **Resultado:** Validação de padrão complexo bem-sucedida
- **Aceita:** Códigos válidos no padrão esperado
- **Rejeita:** Códigos inválidos ou formatações incorretas
- **Observação:** Demonstra computabilidade além de autômatos finitos

5.1.5 Máquina de Mealy

- **Resultado:** Saídas geradas corretamente
- **Observação:** Transições claras com saídas bem definidas
- **Uso Prático:** Ideal para tradutores e processadores de linguagem

5.1.6 Máquina de Moore

- **Resultado:** Estados com saídas associadas funcionando
- **Observação:** Saídas determinísticas por estado
- **Vantagem:** Mais simples para implementação em lógica combinacional

5.1.7 Gramática Livre de Contexto

- **Resultado:** Gera as cadeias esperadas
- **Observação:** Demonstra a capacidade de linguagens livres de contexto em descrever estruturas mais complexas

6 Discussão e Aprendizados

6.1 Conceitos Reforçados

Esta atividade prática permitiu consolidar vários conceitos fundamentais:

1. **Equivalência de Modelos:** Compreender que AFND e AFD são equivalentes em poder de reconhecimento, mas diferem em determinismo;
2. **Hierarquia de Linguagens:** Observar que linguagens regulares (AFD/AFND) são um subconjunto de linguagens livres de contexto (GLC);
3. **Máquinas de Estado com Saída:** Entender a diferença prática entre máquinas de Mealy e Moore, e suas aplicações;
4. **Computabilidade:** Perceber como Máquinas de Turing são o modelo mais geral e podem resolver problemas que autômatos finitos não conseguem.

6.2 Ferramentas Utilizadas

O uso do JFLAP foi instrumental para:

- Visualizar graficamente o funcionamento de autômatos;
- Testar de forma interativa;
- Compreender o fluxo de execução passo a passo;
- Converter entre modelos automaticamente.

6.3 Aplicações Práticas

Os modelos implementados encontram aplicações reais em:

- **Análise Léxica:** Compiladores usam AFDs para tokenizar código fonte;
- **Análise Sintática:** Gramáticas livres de contexto são usadas para parser;
- **Processamento de Linguagem Natural:** Máquinas de estado para análise morfológica;
- **Protocolo de Comunicação:** Máquinas de Mealy/Moore para modelar protocolos;
- **Sistemas Embarcados:** Máquinas de estado para controle lógico.

7 Conclusão

Este relatório demonstrou a implementação prática e validação teórica de diversos modelos computacionais fundamentais. Através da utilização do simulador JFLAP, foi possível:

- Consolidar o entendimento de autômatos finitos (determinísticos e não-determinísticos);
- Demonstrar a equivalência entre AFND e AFD através da conversão automática;
- Implementar máquinas de estado com saída (Mealy e Moore);
- Explorar a potência computacional de Máquinas de Turing;
- Validar o comportamento de cada modelo com exemplos práticos.

Os resultados obtidos confirmam que todos os modelos funcionam conforme esperado, aceitando as linguagens para as quais foram projetados e rejeitando cadeias inválidas. Esta atividade reforça a importância de compreender esses formalismos, pois são a base teórica para praticamente todas as ferramentas computacionais modernas, desde compiladores até sistemas de reconhecimento de padrões.

O conhecimento adquirido nesta atividade é fundamental para o desenvolvimento de compiladores, processadores de linguagem, e para compreender os limites teóricos da computação.

A Arquivos Entregues

A entrega contém os seguintes arquivos:

- `afd_acesso.jff` - Autômato Finito Determinístico
- `afnd_entr_variacoes.jff` - Autômato Finito Não-Determinístico
- `afd_from_afnd_entr_variacoes.jff` - AFD convertido do AFND
- `mt_valida_entrada_codigo.jff` - Máquina de Turing
- `mealy_acesso.jff` - Máquina de Mealy
- `moore_acesso.jff` - Máquina de Moore
- `glc_acesso.txt` - Gramática Livre de Contexto
- `RELATORIO.tex` - Este relatório em LaTeX
- `screenshots/` - Pasta com evidências em imagem dos testes

B Referências Teóricas

- Sipser, M. (2012). *Introduction to the Theory of Computation*. 3rd Edition.
- Linz, P. (2011). *An Introduction to Formal Languages and Automata*. 5th Edition.
- Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2006). *Introduction to Automata Theory, Languages, and Computation*. 3rd Edition.