Relatório de Fractais e L-Systems

Adriel Costa,, Eduardo Marinho, Edson Costa, Fernanda Lustosa, Samuel Morais

Universidade Federal do Rio Grande do Norte - UFRN / Departamento de Informática e Matemática Aplicada - DIMAp

Resumo

Este relatório apresenta a modelagem e implementação de um sistema baseado em autômatos com pilha para o processamento de um *L-System*. A modelagem utilizou conceitos formais de linguagens e autômatos, em que o autômato com pilha foi projetado para reconhecer e processar a gramática do *L-System*, gerando sequências de símbolos que descrevem estruturas recursivas. O sistema desenvolvido converte as sequências processadas em representações visuais, produzindo imagens que demonstram as propriedades geométricas e fractais do *L-System* modelado a partir de um gráfico tartaruga.

1. Introdução

L-Systems (ou Sistemas de Lindenmayer) são modelos formais originalmente propostos para descrever o crescimento de plantas, mas que têm sido amplamente utilizados em diversas áreas, como biologia computacional, modelagem fractal e geração procedural de gráficos. Esses sistemas baseiam-se em regras de substituição e são conhecidos por sua capacidade de representar estruturas recursivas e dinâmicas de forma compacta e eficiente.

No contexto da disciplina DIM0606 - Linguagens Formais e Autômatos, ministrada pelo Dr. Martin Alejandro Musicante, professor do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte, foi proposto o desafio de modelar e implementar um sistema baseado em autômatos com pilha para processar e desenhar curvas geradas a partir de *L-Systems*.

O presente relatório descreve o cumprimento dos seguintes objetivos:

- Identificar um alfabeto suficientemente rico para descrever *L-Systems*, contendo primitivas para empilhar e desempilhar a posição do cursor;
- Propor primitivas que permitam estender *L-Systems* na prática, incluindo diferentes tamanhos e cores de traços, além de ângulos variados de giro para o cursor;
- Desenvolver um algoritmo e um programa que, dado um número natural n e um *L-System* enriquecido, gere a cadeia correspondente à n-ésima reescrita;
- Criar um algoritmo que transforme a cadeia obtida em um programa capaz de utilizar uma biblioteca gráfica de tartaruga para desenhar a curva correspondente;
- Desenvolver um programa que verifique se uma string é, de fato, a n-ésima derivação de um *L-System* dado, utilizando um autômato com pilha;
- Realizar toda a resolução do problema com base em conceitos de autômatos com pilha, garantindo a integração teórica e prática do trabalho.

Este trabalho combina conceitos de linguagens formais, automação computacional e computação gráfica para criar uma solução que explora a riqueza dos *L-Systems*. A modelagem e implementação realizadas são discutidas em detalhe, incluindo os resultados visuais obtidos.

2. Fundamentação Teórica

A construção e análise de sistemas formais são pilares fundamentais da ciência da computação, especialmente no campo de linguagens formais e autômatos. Neste trabalho, unimos dois conceitos fundamentais — *L-Systems* e autômatos com pilha — para explorar suas interseções e aplicações na modelagem computacional e visual.

2.1. L-Systems: Linguagens para Modelagem Recursiva

Propostos pelo botânico Aristid Lindenmayer em 1968, os *L-Systems* foram concebidos para modelar o crescimento de organismos multicelulares, como plantas e fungos. Estes sistemas utilizam gramáticas formais para descrever o comportamento de sistemas dinâmicos e recursivos, sendo definidos por três componentes principais:

1. **Alfabeto**: Um conjunto de símbolos que representam ações ou estados.
2. **Axiomática**: Uma string inicial (ou estado inicial) a partir da qual as reescritas se desenvolvem.
3. **Regras de Produção**: Conjuntos de transformações que aplicam substituições aos símbolos do alfabeto, de forma iterativa.

Os *L-Systems* são frequentemente utilizados para gerar formas fractais e padrões naturais, como ramos de árvores, folhagens e sistemas vasculares. Eles apresentam uma abordagem declarativa para a criação de estruturas complexas, destacando-se por sua simplicidade e expressividade.

2.2. Autômatos com Pilha: Estruturas de Computação Hierárquica

Os autômatos com pilha representam uma classe de máquinas teóricas que estendem os autômatos finitos ao adicionar uma estrutura de pilha à sua funcionalidade. Essa adição permite o processamento de linguagens sensíveis ao contexto, como aquelas utilizadas na modelagem de *L-Systems*.

Um autômato com pilha é composto pelos seguintes elementos:

1. **Estados**: Um conjunto finito que define os possíveis ”momentos”da execução da máquina.
2. **Alfabeto de Entrada**: Conjunto de símbolos que a máquina pode processar como entrada.
3. **Alfabeto da Pilha**: Conjunto de símbolos que podem ser armazenados na pilha.
4. **Função de Transição**: Regras que definem como a máquina muda de estado, manipula a pilha e lê a entrada.

Os autômatos com pilha são especialmente úteis em aplicações que exigem memória dinâmica e hierárquica, como análise sintática de linguagens de programação, processamento de linguagens naturais e, neste caso, a execução de *L-Systems*.

2.3. Conexão entre L-Systems e Autômatos com Pilha

Embora os *L-Systems* sejam tradicionalmente associados a gramáticas de substituição, sua modelagem computacional pode se beneficiar do uso de autômatos com pilha. A pilha atua como uma memória dinâmica que armazena estados intermediários, possibilitando operações como empilhar e desempilhar coordenadas ou atributos do sistema, essenciais para:

- Controle do Cursor: Manipulação de posições e ângulos em sistemas gráficos baseados em tartaruga.
- Atributos Dinâmicos: Alteração de características como cores e espessura de traços durante a execução.

Ao combinar as capacidades de processamento hierárquico dos autômatos com pilha com as propriedades declarativas dos *L-Systems*, é possível criar um modelo robusto para interpretar e gerar estruturas recursivas, além de verificar a conformidade das strings geradas.

3. Metodologia

3.1. Alfabeto do L-System

Para o desenvolvimento do sistema, foi definido um alfabeto suficientemente rico para descrever *L-Systems* e atender aos requisitos do projeto. O alfabeto utilizado é o seguinte:

$$\Sigma = \{F, f, G, g, [,], r, g, b, +, -, T, t\}$$

Cada elemento do alfabeto possui uma função específica no contexto do *L-System*, como controle do cursor, manipulação da pilha ou ajuste de atributos gráficos. Essa definição visa proporcionar flexibilidade e expressividade suficientes para criar representações gráficas complexas.

Além disso, as regras de derivação dos L-Systems não foram fixadas no código-fonte, mas sim configuradas pelo usuário no momento da execução. Isso possibilita

maior personalização e adaptabilidade, permitindo a exploração de diferentes formas e padrões.

A tabela a seguir detalha o significado e a função de cada símbolo no alfabeto:

Tabela 1: Descrição dos símbolos do alfabeto utilizado

Símbolo	**Descrição**
F	Move o cursor para frente, desenhando uma linha.
f	Move o cursor para frente sem desenhar uma linha.
G	Similar a **F**, mas com um comportamento gráfico específico para distinção em algumas curvas.
g	Similar a **f**, mas sem produzir traços.
[	Empilha o estado atual do cursor (posição e orientação).
]	Desempilha o estado do cursor, retornando ao estado anterior.
r	Ajusta a cor do traço para vermelho.
g	Ajusta a cor do traço para verde.
b	Ajusta a cor do traço para azul.
+	Gira o cursor para a direita em um ângulo definido pelo usuário.
-	Gira o cursor para a esquerda em um ângulo definido pelo usuário.
T	Ajusta a espessura do traço para um valor maior.
t	Ajusta a espessura do traço para um valor menor.

Esse alfabeto foi escolhido para garantir um equilíbrio entre simplicidade e funcionalidade, permitindo a modelagem de *L-Systems* tanto básicos quanto avançados, com capacidade de manipulação gráfica integrada.

3.2. Geração de Cadeias

A seguir, detalhamos as principais fases do processo de geração de cadeias no sistema:

- **Entrada de Dados e Validação**: O sistema recebe como entrada o número de iterações, o ângulo de rotação e o axioma inicial, que serve como ponto de partida para o L-System. Antes de iniciar a produção de cadeias, os símbolos contidos

no axioma são validados em relação ao alfabeto definido, garantindo que apenas símbolos reconhecidos sejam processados.

- **Produção Iterativa de Cadeias**: A partir do axioma inicial, o algoritmo substitui iterativamente as variáveis presentes na cadeia com base nas regras de produção fornecidas pelo usuário. Esse processo é realizado para o número de iterações especificado, permitindo a construção progressiva de cadeias mais complexas a cada passo.

- **Renderização da Cadeia Final**: Após a última iteração, a cadeia resultante é traduzida em uma representação gráfica utilizando a biblioteca turtle. Cada símbolo da cadeia é mapeado para uma ação específica, como movimentação do cursor, mudança de cor ou alteração do ângulo, resultando na renderização da curva descrita pelo *L-System*.

3.3. Checagem de Cadeias

O sistema também implementa a funcionalidade de verificar se uma cadeia fornecida corresponde à *n*-ésima derivação de um *L-System*. Esse processo segue os passos descritos a seguir:

- **Preparação dos Dados**: O programa recebe como entrada o axioma inicial, a cadeia esperada, as regras de produção e o número de iterações. Essas informações servem como base para realizar a validação.

- **Aplicação das Regras e Recursão**: Utilizando um autômato com pilha, o sistema processa recursivamente as regras de produção a partir do axioma. Em cada etapa, as variáveis da cadeia são substituídas pelas correspondentes definições nas regras até que o número de iterações especificado seja alcançado.

- **Finalização e Verificação**: Ao término do processamento, o sistema compara a cadeia gerada com a cadeia fornecida como entrada. A pilha é esvaziada, e a correspondência entre ambas é analisada. Caso a cadeia fornecida corresponda exatamente à *n*-ésima derivação do *L-System*, o sistema retorna um resultado positivo; caso contrário, retorna um resultado negativo.

3.4. Integração com Autômato com Pilha

Uma característica central do projeto foi o uso de um autômato com pilha para controlar tanto a geração quanto a validação das cadeias. A pilha foi utilizada para armazenar estados intermediários e possibilitar a recursividade do processo. Isso permitiu implementar as operações de empilhar e desempilhar de maneira natural, garantindo a compatibilidade com as operações definidas no alfabeto, como [e].

4. Desenvolvimento do Sistema

O sistema foi desenvolvido utilizando a linguagem de programação Python3, empregando algumas bibliotecas externas que facilitaram sua implementação. A biblioteca `turtle` foi usada para a renderização gráfica do fractal a partir de uma cadeia de caracteres, enquanto `sys` permitiu a interação com o terminal para recepção de parâmetros do usuário. Além disso, o módulo `namedtuple`, da biblioteca `collections`, foi utilizado para armazenar o estado do programa durante a renderização.

4.1. Estrutura do Projeto

O projeto foi organizado em um diretório chamado `src`, contendo cinco arquivos principais desenvolvidos em Python3:

- `LSystem.py`: Responsável pela lógica principal do *L-System*, incluindo validação de cadeias geradas, aplicação das regras de produção ao axioma para gerar novas cadeias e verificação da consistência de cadeias fornecidas com o *L-System* definido.
- `alphabet.py`: Define o alfabeto do *L-System*, mapeando caracteres para ações específicas da biblioteca `turtle`.
- `main.py`: Centraliza a interação com o usuário, permitindo a leitura de parâmetros fornecidos via terminal e a solicitação de informações, como número de iterações, regras de reescrita, ângulo e axioma. Esse arquivo também determina a funcionalidade desejada (validação ou renderização) e gerencia o fluxo da aplicação.

- `renderer.py`: Implementa a renderização gráfica do *L-System*, utilizando a biblioteca `turtle`. Essa classe inclui operações como salvamento e restauração do estado da ”tartaruga”e desenho dos caracteres definidos no alfabeto.

- `test.py`: Inclui testes simples para validar partes específicas do código desenvolvido.

4.2. Validação de Strings

A validação de strings é uma funcionalidade essencial do sistema e verifica se uma determinada cadeia de caracteres corresponde ao resultado esperado de um *L-System* após um número específico de reescritas. Para isso, o usuário deve fornecer algumas entradas, como o número de iterações, o axioma inicial baseado no alfabeto definido, a string a ser validada e as regras de derivação no formato `char->string`.

O processo inicia com a leitura dos argumentos passados na linha de comando, indicando que a operação de validação deve ser realizada. Em seguida, os parâmetros necessários, como número de iterações, axioma e string, são coletados, e as regras de reescrita são registradas. Por fim, a função `check()` é chamada no arquivo `LSystem.py`, comparando a string fornecida com o resultado da reescrita do axioma de acordo com as regras e iterações especificadas.

`Renderização de Fractais`

A renderização gráfica do *L-System* é realizada por meio de comandos executados no terminal, onde o usuário especifica os parâmetros necessários, como o número de iterações, o ângulo de rotação e o axioma inicial. Além disso, é possível definir diversas regras de derivação no formato `char->string`.

O processo de renderização inicia com a leitura dos argumentos fornecidos na linha de comando, indicando que essa funcionalidade será utilizada. O axioma é validado para garantir sua conformidade com o alfabeto definido. Caso a validação seja bem-sucedida, a função `apply_production_rules()` é chamada para construir o *L-System* final, que será exibido e renderizado na tela.

A biblioteca `turtle` desempenha um papel crucial nesse processo, traduzindo os caracteres do alfabeto em ações gráficas. Durante a execução, estados da ”tartaruga”são

armazenados e recuperados de uma pilha, permitindo a criação de estruturas fractais complexas.

5. Resultados

Os resultados obtidos a partir da implementação do *L-System* demonstram a versatilidade e riqueza visual proporcionada por diferentes configurações de axiomas, ângulos e regras de produção. A seguir, apresentam-se as imagens geradas com base nas especificações de entrada:

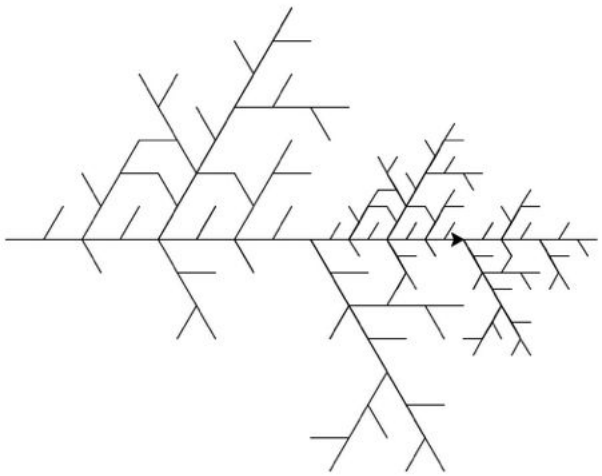

Figura 1: Exemplo de fractal gerado com L-System.

A **Figura 1** ilustra o resultado de um *L-System* configurado com 3 iterações, ângulo de 60 graus e axioma "Ff". As regras de produção utilizadas foram: $F \rightarrow F[+F]F[-F][F]$ e $f \rightarrow f[+f]f[-f][f]$. Essa configuração gera uma estrutura altamente ramificada, reminiscentes de padrões encontrados em sistemas biológicos, como galhos de árvores.

Na **Figura 2**, observa-se o resultado de 4 iterações, com um ângulo de 90 graus e axioma "F". As regras de produção $F \rightarrow rf[+bF] - gF$ conferem ao fractal um

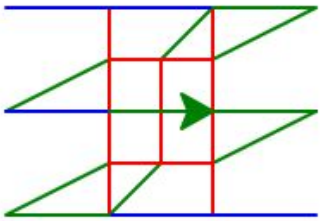

Figura 2: Exemplo de fractal gerado com L-System.

padrão de desenvolvimento assimétrico, evidenciando o impacto de modificações nos símbolos utilizados e nas regras de derivação.

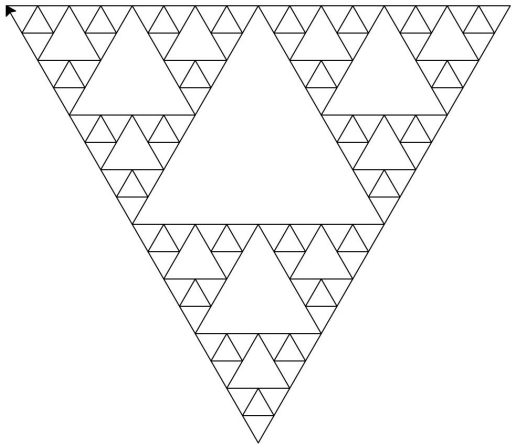

Figura 3: Exemplo de fractal gerado com L-System.

A **Figura 3** apresenta o padrão obtido a partir de 4 iterações, com ângulo de 120 graus e axioma "$F-G-G$". Neste caso, as regras de produção $F \rightarrow F-G+F+G-F$ e $G \rightarrow GG$ resultam em um fractal com uma estrutura repetitiva e simétrica,

frequentemente associada a triângulos fractais.

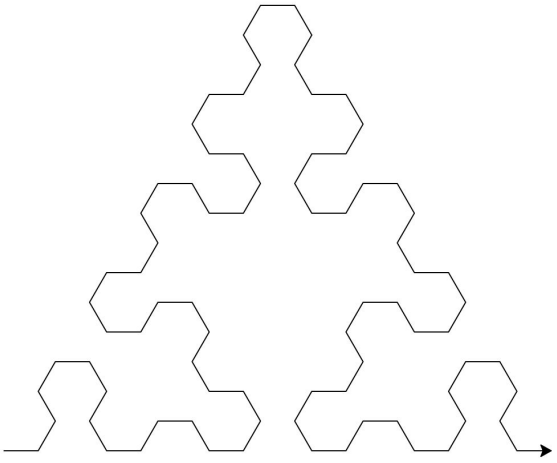

Figura 4: Exemplo de fractal gerado com L-System.

Por fim, a **Figura 4** exibe o resultado gerado com 4 iterações, ângulo de 60 graus e axioma "F". Utilizando as regras de produção $F \rightarrow G-F-G$ e $G \rightarrow F+G+F$, obteve-se uma estrutura intrincada e simétrica, com padrões que remetem a formações geométricas complexas.

Esses resultados evidenciam a capacidade do $L-System$ de gerar formas variadas e intrincadas, que são amplamente exploradas em estudos de simulação natural e geração de padrões visuais complexos. As figuras destacam a sensibilidade dos fractais às condições iniciais e às regras estabelecidas, refletindo as propriedades fundamentais dos sistemas dinâmicos.

6. Conclusão

Os modelos teóricos utilizados demonstraram ser adequados para alcançar os objetivos propostos, e os códigos que implementaram esses modelos funcionaram conforme o esperado. Assim, o trabalho foi realizado com êxito, validando a eficácia das abordagens empregadas e a consistência dos resultados obtidos.

7. Tabela de Participação

Tabela de Participação	
Nome	**Participação (de 0 a 10)**
Adriel Costa	10
Eduardo Marinho	10
Edson Costa	10
Fernanda Lustosa	10
Samuel Morais	10

Tabela 2: Participação dos membros no projeto.

8. Referências

WIKIPEDIA. L-system. Disponível em: https://en.wikipedia.org/wiki/L-system. Acesso em: 02 dez. 2024.

PRUSINKIEWICZ, Przemyslaw; LINDENMAYER, Aristid. The Algorithmic Beauty of Plants. 1. ed. New York: Springer-Verlag, 1990. Cap. 1, p. 1-18.