UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

OBTENÇÃO DE GRAMÁTICAS *L-SYSTEMS* A PARTIR DE CADEIAS DE CARACTERES

EDMAR SANTOS

ORIENTADORA: PROFA. DRA. REGINA CÉLIA COELHO

PIRACICABA, SP

2008

UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

OBTENÇÃO DE GRAMÁTICAS *L-SYSTEMS* A PARTIR DE CADEIAS DE CARACTERES

EDMAR SANTOS

ORIENTADORA: PROFA. DRA REGINA CÉLIA COELHO

Dissertação apresentada ao Mestrado em Ciência da Computação, da Faculdade de Ciências Exatas e da Natureza, da Universidade Metodista de Piracicaba – UNIMEP, como requisito para obtenção do Título de Mestre em Ciência da Computação.

PIRACICABA, SP

2008

Santos, Edmar

Obtenção de gramáticas *L-systems* a partir de cadeias de caracteres / Edmar Santos. -- Piracicaba,SP: UNIMEP / Programa de Pós-Graduação em Ciência da Computação, 2008.
78p.

Orientação: Profa. Dra. Regina Célia Coelho
Dissertação (Mestrado) - Programa de Pós-Graduação em Ciência da Computação, Universidade Metodista de Piracicaba/UNIMEP.

1. *L-system*; 2. Problema Inverso; 3. Problema Inverso de Lindenmayer; 4. Inferência Gramatical; I. Santos, Edmar; II. Título.

OBTENÇÃO DE GRAMÁTICAS *L-SYSTEMS* A PARTIR DE CADEIAS DE CARACTERES

AUTOR: EDMAR SANTOS

ORIENTADORA: PROFA. DRA. REGINA CÉLIA COELHO

Dissertação de Mestrado apresentada em 20 de Fevereiro de 2008, à Banca Examinadora constituída pelos Professores:

_____
Profa. Dra. Regina Célia Coelho
UNIMEP

_____
Prof. Dr. Marcio Merino Fernandes
UNIMEP

_____
Prof. Dr. Roberto Marcondes César Jr.
IME-USP

DEDICATÓRIA

À

Deus pelo amor, graça e misericórdia estendida a nós;

À

Minha esposa Lina pelo apoio e sacrifício incondicional, mesmo diante das adversidades que nos assolaram no último ano;

Ao meu filho Lucas que suportou ficar sem o papai por tantas vezes, e ao meu filho Leandro que está para chegar.

AGRADECIMENTOS

À professora Regina Célia Coelho pela sabedoria disseminada durante as orientações no desenvolvimento deste trabalho;

Aos professores da UNIMEP pelos conhecimentos compartilhados nas disciplinas;

Ao UNASP-SP pela confiança exercida e pelo investimento dispensado;

Aos amigos Rubem César Tavares e Hélvio Carvalho de Araújo pelas oportunidades e incentivos;

Aos amigos Carlos, Clodonil e Wilson pelas batalhas travadas em união e apoio.

RESUMO

O grande problema das abordagens que empregam o uso dos *L-systems* está justamente na determinação de uma gramática formal que represente apropriadamente o modelo desejado. Na teoria dos *L-systems* esse problema é conhecido como o Problema Inverso de Lindenmayer, um subcaso dos processos de inferência gramatical. Este trabalho apresenta uma proposta para solucionar o Problema Inverso de Lindenmayer nas classes de gramáticas livres de contexto e determinísticas, além de apresentar estudos sobre a resolução desse problema nas gramáticas não determinísticas. A abordagem desta proposta apresenta uma metodologia que consegue obter uma regra *L-system* a partir de uma cadeia de caracteres representante do estágio de desenvolvimento de um objeto qualquer. As cadeias utilizadas nos testes são sintetizadas a partir de gramáticas conhecidas, porém são tratadas como de origem desconhecida para assegurar a imparcialidade da metodologia. A idéia aqui apresentada consiste na regressão do crescimento da cadeia analisada por um algoritmo construído com base nas relações de crescimento obtidas a partir de cadeias geradas por gramáticas determinísticas conhecidas. Nos testes realizados, todas as cadeias submetidas no algoritmo puderam ser revertidas em uma regra *L-system* idêntica à regra original utilizada na síntese da cadeia. Também é interessante notar que a obtenção destas regras ocorreu praticamente em tempo real em todas as gramáticas testadas. Os testes realizados nos estudos das gramáticas não determinísticas utilizadas como exemplos também resultaram em regras exatas.

PALAVRAS-CHAVE: *L-system*, Problema Inverso, Problema Inverso de Lindenmayer, Inferência Gramatical

ABSTRACT

The biggest problem with approaches that employ the use of L-systems is exactly the determining a formal grammar that appropriately represents the desired model. In the L-systems theory this problem is known as the Inverse Problem of Lindenmayer, a subset of processes of grammatical inference. This study proposes a solution for the Inverse Problem of Lindenmayer in context-free and deterministic grammatical classes, in addition to the initial studies on the resolution of this problem in non-deterministic grammars. The approach of this proposal presents a methodology that can obtain an L-system rule from a string representing the development stage of any object. The strings used in the tests were obtained from known grammars, however, they are dealt with as having an unknown origin to assure the impartiality of the methodology. The idea presented here consists in the growth regression of the string analyzed by an algorithm built based on the relations of growth obtained from string generated by known deterministic grammars. In the tests carried out, all the strings submitted to the proposed algorithm could be reverted to an L-system rule identical to the original rule used in the synthesis of the string. It is also interesting to observe that the obtaining of these rules occurred practically in real time with the tested grammars. Tests performed in these studies of the non-deterministic grammars used as examples also resulted in exact rules.

KEY-WORDS: *L-system*, Inverse Problem, Inverse Problem of Lindenmayer, Grammar Inference

SUMÁRIO

LISTA DE QUADROS

LISTA DE FIGURAS

1 INTRODUÇÃO

Em 1968, é concebida pelo biólogo húngaro Aristid Lindenmayer uma teoria matemática capaz de formalizar o desenvolvimento celular de pequenos organismos multicelulares (LINDENMAYER, 1968) baseada nas gramáticas de Chomsky e sistemas de reescrita (PRUSINKIEWICZ, 1986a). Em sua homenagem, essa teoria ficou conhecida como Sistemas de Lindenmayer ou simplesmente *L-systems*.

A abordagem dos sistemas de Lindenmayer, originalmente tratava dos aspectos topológicos no desenvolvimento de pequenos filamentos de organismos multicelulares. Inicialmente, toda representação gráfica era feita manualmente por meio de figuras, indicando o processo de desenvolvimento desses organismos.

A introdução de mecanismos mais avançados para representações gráficas dos processos de desenvolvimento formalizados pelos *L-systems* foram surgindo mais tarde. Um dos primeiros trabalhos nessa direção é apresentado por FRIJTERS e LINDEMAYER (1974). Neste trabalho é proposto um modelo computacional capaz de reproduzir o processo de crescimento e florescência de plantas herbáceas.

Nessa mesma época, HOGEWEG e HESPER (1974) propuseram a forma de interpretação da cadeia de caracteres resultante dos *L-systems* como funções que atuariam na morfologia do modelo durante sua reconstrução.

Posteriormente surgiram ainda outras aplicações para representações gráficas dos processos de desenvolvimento evoluídos pelos sistemas de Lindenmayer, tais como:

- SZILARD e QUINTON (1979), que mostraram o uso dos *L-systems* na geração de fractais com rigorosa definição geométrica;
- SIROMONEY e SUBRAMANIAN (1983), que direcionaram seu trabalho na geração das chamadas curvas de Peano ou *spacefilling-curves* (SAGAN, 1994);

- AONO e KUNII (1984) e SMITH (1984), que também aplicaram os *L-systems* na geração de fractais, além da síntese de imagens realísticas de árvores e plantas menores;
- PRUSINKIEWICZ (1986a), que introduz a interpretação gráfica da cadeia de caracteres gerada pelos *L-systems* baseada em analogias à geometria da tartaruga proveniente da linguagem LOGO (PAPERT, 1986). Mais tarde, essa abordagem tornou-se um modelo clássico para a interpretação gráfica dos sistemas de Lindenmayer.

Desde então, a potencialidade desses sistemas ficou evidente na forma como muitas linhas de pesquisas se apossaram desses recursos e lançaram novas abordagens:

- BOER (1991) aplicou os *L-systems* na análise do processo de desenvolvimento de camadas de células;
- BRONS (1995) aponta os *L-systems* para geração de retas digitais;
- COWELL (1995) promove o reconhecimento de padrões na identificação de placas de automóveis;
- HOLLIDAY (1995) e SAMALL, PETERSON e HOLLIDAY (2002), que discutem o reconhecimento de formas vegetais como folhas, plantas menores e árvores em visão computacional;
- PRUSINKIEWICZ (1986b) apresenta uma interessante aplicação na composição de arranjos musicais a partir da interpretação gráfica da curva de Hilbert. O tamanho dos segmentos traçados horizontalmente na imagem construída controla o período de duração da nota musical. Já o posicionamento no eixo y do plano cartesiano corresponde a uma nota musical;
- AHO et al. (1997) introduz um método para procura de estruturas de redes neurais;
- HORNBY e POLLACK (2001) usa os *L-systems*, juntamente com um algoritmo evolucionário, para gerar criaturas virtuais para ambientes de simulação;
- HAMILTON (1993), MCCORMICK e MULCHANDANI (1994), KALAY, PARNAS e SHAMIR (1995), além de COELHO e JAQUES (2003), empregaram *L-systems* na geração de neurônios artificiais.

1.1 MOTIVAÇÃO

Uma das maiores dificuldades existentes em grande parte das abordagens que empregam o uso dos *L-systems* está em determinar o formalismo que represente, convenientemente, o processo de desenvolvimento da estrutura desejada. Codificar esse processo em uma notação *L-system* para que se permita sua posterior reprodução é a grande questão. Um problema com essas características pode ser compreendido como um caso do **problema inverso**, teoria amplamente estudada em áreas como geofísica (MENKE, 1989; PARKER, 1994), física (BERTERO; BOCCACCI, 1998), pesquisas em imagens médicas (NATTERER; WÛBBELING, 2001; UHLMANN, 2003), além de outras. De forma geral, trata-se de um problema em que se pretende inferir a representação de um modelo baseado na observação e análise de dados pertencentes ao mesmo sistema desse modelo (CHALMOND, 2003; RAMM, 2005; TARANTOLA, 2005, 2006). Na teoria dos *L-systems,* esse problema é conhecido como o **problema inverso de Lindenmayer**.

1.2 OBJETIVO

O presente trabalho esteve, justamente, focado em procurar uma metodologia capaz de auxiliar na busca dessa notação formal. O objetivo desta pesquisa foi identificar procedimentos que permitam obter a descrição formal de um processo evolutivo. A idéia consistiu em reverter uma cadeia de caracteres descritora de um determinado estado de desenvolvimento de uma estrutura auto-similar, em uma regra *L-system*, de modo que essa regra permita a reprodução desse objeto. Assumiu-se que essas cadeias já tenham sido obtidas por algum processo. Assim, não foi intenção, neste trabalho, discutir os métodos de obtenção dessas cadeias. Além disso, neste trabalho, foram consideradas gramáticas *L-systems* Livres de Contexto e Determinísticas. Também foram realizados estudos sobre a obtenção de gramáticas *L-systems* Livre de Contexto e Não Determinísticas a partir de amostras geradas também por gramáticas conhecidas. A intenção neste trabalho não foi realizar estudos exaustivos em relação à obtenção da

gramática não determinística a partir da cadeia de caracteres, mas apenas obter resultados iniciais e que indiquem um caminho para uma possível solução futura para qualquer cadeia de caracteres obtida utilizando este tipo de gramática. Os processos de investigação desses procedimentos foram realizados com base nas principais características dos *L-systems*, como auto-similaridade e reescrita paralela, e não como ocorre nos processos de inferências gramaticais tradicionais realizados pela enumeração das soluções possíveis ou reconstrução gramatical.

1.3 CONTRIBUIÇÕES

A contribuição direta deste trabalho, além de constituir uma nova abordagem sobre resolução do problema inverso de Lindenmayer para uma subclasse de gramáticas *L-systems*, foi prover subsídios para obtenção de regras *L-systems* que serão utilizadas na reconstrução de estruturas neurais em ambientes de realidade virtual para interação e visualização.

1.4 ORGANIZAÇÃO

A organização deste trabalho é feita da seguinte forma: no capítulo 2, é apresentada a revisão bibliográfica com a exposição resumida de conceitos sobre linguagens formais, bem como o estado da arte em relação à inferência gramatical e ao problema inverso de Lindenmayer; no capítulo 3 e 4 é delineada a proposta deste trabalho com a apresentação da metodologia empregada, seguida dos resultados obtidos e a discussão sobre eles; no capítulo 5 são apresentadas as conclusões, e, ao final, são listadas as referências consultadas para a elaboração deste trabalho.

2 REVISÃO DA LITERATURA

Neste capítulo são apresentados, de forma geral, os conceitos e fundamentos teóricos que foram necessários ao desenvolvimento deste trabalho subdivididos em tópicos menores.

Primeiramente, na seção 2.1, são apresentados os principais conceitos sobre as linguagens formais usados pela Ciência da Computação baseados nas obras de COHEN (1997), SUDKAMP (1997) e MENEZES (2001).

Já na seção 2.2, são apresentados o formalismo e as definições sobre os sistemas de Lindenmayer baseados em PRUSINKIEWICZ e LINDENMAYER (1990) e outros trabalhos (LINDENMAYER, 1968; PRUSINKIEWICZ, 1986a), além de sua correlação com Hierarquia de Chomsky dentro da classificação das linguagens formais.

Por fim, na seção 2.3 é feito um levantamento teórico sobre a questão da inferência gramatical, e, na seção 2.4, são apresentados alguns trabalhos que revelam o estado da arte sobre problema inverso de Lindenmayer. Ambas as seções estão referenciadas localmente.

2.1 LINGUAGENS FORMAIS E MÁQUINAS DE ESTADO

Originário da Teoria das Linguagens Formais, que estava direcionada ao desenvolvimento das teorias das linguagens naturais em 1950, as linguagens formais tornaram-se presentes na Ciência da Computação por revelarem-se apropriadas ao estudo das linguagens artificiais. A finalidade das linguagens formais está, principalmente, relacionada às questões sintáticas das linguagens (MENEZES, 2001).

Dessa forma, compreender o formalismo de uma linguagem implica fazer algumas definições necessárias como a introdução dos símbolos que poderão ser utilizados; a definição de um alfabeto, palavras e linguagens,

além da própria definição das gramáticas e sua classificação. Segue na seqüência a definição resumida desses conceitos.

Definição 1 – Símbolo

Um símbolo é uma entidade abstrata e por si só sem interpretação.

Comumente, caracteres como letras (a, b, c, etc) e dígitos ($1, 2, 3, etc$) são utilizados como símbolos, além de outras marcas especiais $(+, -, [,], \wedge, @, \#, \$, etc)$.

Definição 2 - Alfabeto

Um alfabeto é um conjunto finito de símbolos, geralmente, denotado pela letra grega maiúscula $\sum$ (sigma).

Vale ressaltar que um conjunto vazio também é considerado um alfabeto. $\sum_1 = \{\}$ e $\sum_2 = \{a, b, c, 1, 2\}$ são exemplos de alfabetos.

Definição 3 - Palavra, cadeia de caracteres ou sentença

Uma palavra, cadeia de caracteres ou sentença é qualquer seqüência finita dos símbolos existentes no alfabeto. É, freqüentemente, denotada por w. Uma palavra sem símbolos é denominada “palavra vazia” e, geralmente, é representada por ε (*épsilon*).

Dado um alfabeto $\sum$**,** então a representação $\sum^*$ denota o conjunto de todas as palavras possíveis neste alfabeto, e a representação $\sum^+$ denota o conjunto de todas as palavras possíveis em $\sum$ excetuando-se ε.

O tamanho de uma cadeia de caracteres w pode ser denotado como $|w|$ e indica o número de símbolos presentes na cadeia. Por exemplo, o tamanho da palavra $w_1 = abc$ é representado por $|w_1| = 3$.

Definição 4 - Linguagem

Uma linguagem formal é o conjunto de palavras possíveis que podem ser formadas sobre um alfabeto. É representada por vezes pela letra L.

Dado um alfabeto $\sum = \{0,1\}$, a linguagem L sobre $\sum^*$ é $\{ \varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, ... \}$.

Definição 5 - Gramática

Uma gramática é uma quádrupla ordenada $G = \{ V, T, P, S\}$ em que:

V**:** é o conjunto de símbolos chamados de variáveis ou não-terminais, normalmente, designados por maiúsculas como *S, A, B, F*, etc.

T**:** é o conjunto de símbolos chamados de constantes ou terminais, normalmente, designados por minúsculas e outros símbolos como *0, 1, +, -, a, b,* etc.

P**:** é o conjunto finito de pares (α, β) chamados de regras de produção ou derivação, em que $\alpha \in (V \cup T)^+$ é chamado de predecessor e $\beta \in (V \cup T)^*$ chamado de sucessor. A regra de produção é representada na forma $\alpha \to \beta$ ou $p_k : \alpha \to \beta$**,** sendo k um identificador inteiro para a regra de produção e lê-se que α produz ou deriva β. As regras de produção são aplicadas de forma seqüencial, uma por vez, e a aplicação de cada regra é representada pelo símbolo $\Rightarrow$ que também pode referenciar o número i de derivações sucessivas na forma $\overset{i}{\Rightarrow}$.

S**:** é o axioma, o primeiro elemento sobre o qual se dará a aplicação das regras de produção, sendo $S \in V^+$.

Em relação às regras de produção, uma gramática pode ser apresentada como:

- **Determinística:** quando houver apenas uma possibilidade de derivação para cada regra de produção, podendo ser denotado por $\alpha_1 \to \beta_1,\ \alpha_2 \to \beta_2,\ \alpha_3 \to \beta_3, \ldots,\ \alpha_n \to \beta_n$. O item (a) do Quadro 1 é um exemplo de uma gramática determinística;
- **Não determinística:** quando para uma mesma regra de produção houver mais de uma possibilidade de reescrita, podendo ser denotada por $\alpha \to \beta_1,\ \alpha \to \beta_2,\ \alpha \to \beta_3, \ldots, \alpha \to \beta_n$ ou $\alpha \to \beta_1 \mid \beta_2 \mid \beta_3 \mid \ldots \mid \beta_n$. O item (b) do Quadro 1 é um exemplo de uma gramática não determinística;

Além disso, quando as regras de produção de uma gramática estão condicionadas a pesos probabilísticos de ocorrência ela é dita uma **gramática estocástica**. Nesse caso, as regras poderão estar na forma $A \to \beta_i : p_i$ na qual $A \in V$ e $\beta \in (V \cup T)^*$, sendo $0 \leq p \leq 1$ para cada regra $(A,\ \beta,\ p)$, além de $\sum p_i = 1$. O item (c) do Quadro 1 exemplifica uma gramática estocástica.

Quadro 1 - Exemplo de gramática a) determinística, b) não determinística e c) estocástica

a)	b)	c)
$V = \{S, A, B\}$ $T = \{a, b\}$ $P = \{S \to A,$ $\quad A \to ABa$ $\quad B \to b\}$	$V = \{S, A\}$ $T = \{a, b\}$ $P = \{S \to AA,$ $\quad A \to Aa$ $\quad A \to Ab\}$ ou $V = \{S, A\}$ $T = \{a, b\}$ $P = \{S \to AA,$ $\quad A \to Aa \mid Ab\}$	$V = \{S, A\}$ $T = \{a, b\}$ $P = \{S \to AA,$ $\quad A \to Aa : 0.75$ $\quad A \to Ab : 0.25\}$

Considerando-se uma gramática $G = \{V, T, P, S\}$, a linguagem gerada por essa gramática é denotada por $L(G)$ e representa o conjunto de todas as palavras de símbolos terminais que podem ser produzidas a partir do axioma S por meio das regras de derivação. Gramáticas que geram as mesmas linguagens são ditas gramáticas equivalentes.

Uma gramática é considerada um formalismo gerador ou axiomático, pois permite determinar se uma palavra pode ser gerada ou não por essa gramática.

Paralelo à definição da gramática G_1 apresentada no Quadro 2 da subseção 2.1.2, podem ser observados sete níveis de produções seqüenciais, derivando a palavra $ababaa$. Na Figura 1, é apresentado o processo de derivação dessa palavra, evidenciando a aplicação seqüencial das

regras de produção, pelos indicadores entre parênteses. O processo de derivação apresentado representa apenas uma das possíveis seqüências, pois esta palavra pode ser gerada por diferentes seqüências de produções nesta mesma gramática.

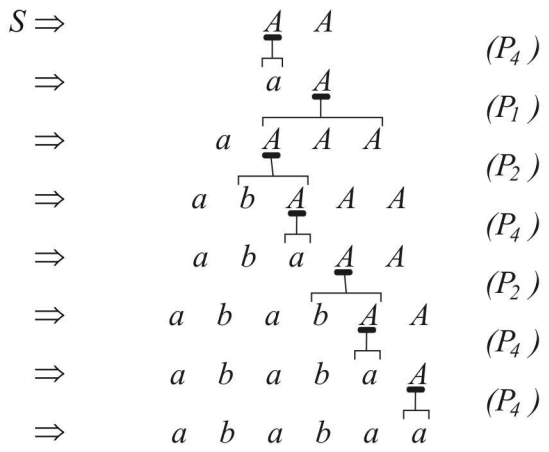


Figura 1 - Processo de derivação seqüencial da gramática G_1

2.1.1 MÁQUINA DE ESTADO OU RECONHECEDOR

Uma máquina de estado ou máquina abstrata é um formalismo baseado em estados. Esses estados são alterados mediante a atuação de instruções primitivas fornecida à máquina. É dita um formalismo reconhecedor, pois permite verificar se uma dada palavra é reconhecida por ela ou não. Dentre os principais reconhecedores cita-se o **Autômato Finito**, **Autômato com Pilha** e **Máquina de Turing**.

Apenas para ilustrar este trabalho, na seqüência é apresentada a definição formal de um Autômato Finito, seguido de um exemplo de autômato que reconhece a linguagem $L_1 = \{w \mid w \text{ possui } aa \text{ ou } bb \text{ como subpalavra}\}$. Trata-se de um exemplo obtido de MENEZES (2001, p 34 e 35).

Autômato Finito: um autômato finito é normalmente definido por uma 5-upla $M = (\sum, Q, \delta, q_0, F)$, na qual:

$\sum$: representa o alfabeto da linguagem utilizada;

Q: representa o conjunto finito de estados;

δ: representa a função de transição de estados;

q_0: representa o estado inicial;

F: representa o conjunto de estados finais possíveis;

O autômato finito $M_1 = (\{a,b\}, \{q_0,q_1,q_2, q_f\}, \delta_1, q_0, \{q_f\})$, que reconhece a linguagem L_1 pode ser representado por um grafo, conforme Figura 2. Nessa figura as setas indicam as funções de mudanças de estado definidas para essa máquina. M_1 reconhece qualquer palavra gerada sobre o alfabeto $\{a,b\}$ que contenha a seqüência aa ou bb como subpalavra.

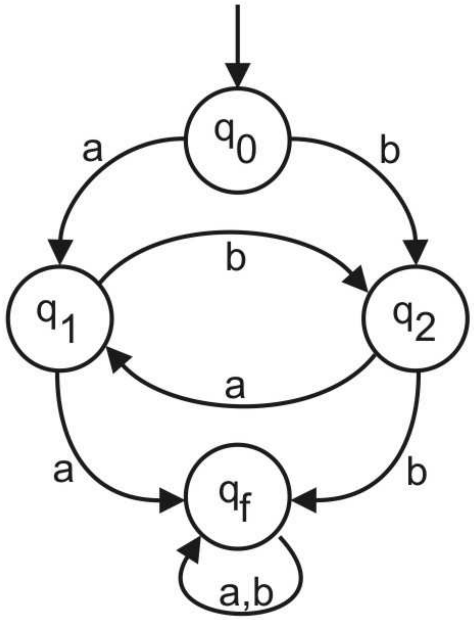


Figura 2 - Grafo do Autômato Finito M_1 baseado em MENEZES (2001, p. 35)

2.1.2 HIERARQUIA DE CHOMSKY

Em 1956, Noam Chomsky descreveu o que hoje é conhecido como Hierarquia de Chomsky. Trata-se de uma classificação das linguagens formais de acordo com as regras de produção e autonomia que as gramáticas impõem sobre a construção da linguagem. Essa hierarquia conta com quatro classes de gramáticas: **Gramáticas Tipo 0, Enumeráveis Recursivamente ou**

Irrestritas; Gramáticas Tipo 1 ou Sensíveis ao Contexto; Gramáticas Tipo 2 ou Gramáticas Livres de Contexto e as **Gramáticas Tipo 3 ou Gramáticas Regulares**. A relação de pertinência entre essas classes pode ser observada na Figura 3, obtida com base em MENEZES (2001, p. 154).

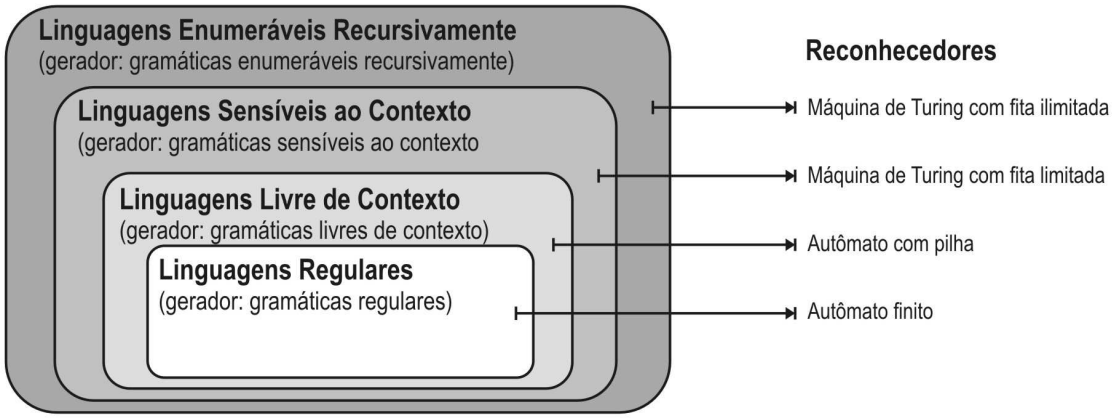


Figura 3 - Hierarquia de Chomsky e máquinas reconhecedoras

Tipo 0 ou Enumeráveis Recursivamente ou Gramáticas Irrestritas: trata-se da classe de todas as possíveis gramáticas reconhecidas por uma máquina de Turing. Nessa classe não há qualquer tipo de restrição, exceto que as regras de produção estejam na forma $\alpha \rightarrow \beta$**,** sendo que $\alpha \neq \varepsilon$.

As linguagens geradas por essa classe de gramática são chamadas de Linguagens Enumeráveis Recursivamente e são reconhecidas por uma máquina de Turing com fita ilimitada.

No Quadro 2, aparece a definição da gramática G_2 como exemplo de gramática Tipo 0 juntamente com cinco níveis de produção, derivando a palavra $aaabbbccc$.

Tipo 1 ou Gramáticas Sensíveis ao Contexto: trata-se da classe de gramáticas em que as regras de produção estão na forma $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$ sendo que α_1, α_2 e $\beta \in (V \cup T)^*$, $\beta \neq \varepsilon$ e $A \in V$. O termo “sensível ao contexto” implica em considerar a existência de α_1 e/ou α_2 em relação ao símbolo A antes de proceder a derivação.

Quadro 2 - Exemplo de gramática a) Livre de Contexto, b) Enumerável Recursivamente, c) Sensível ao Contexto e d) Regular

Definição	Nível	Produção
a) Gramática G_1 $V = \{S, A\}$ $T = \{a, b\}$ $P = \{S \rightarrow AA,$ $P_1 : A \rightarrow AAA$ $P_2 : A \rightarrow bA$ $P_3 : A \rightarrow Ab$ $P_4 : A \rightarrow a\ \}$	n=0	AA
	n=1	aA
	n=2	aAAA
	n=3	abAAA
	n=4	abaAA
	n=5	ababAA
	n=6	ababaA
	n=7	ababaa
b) Gramática G_2 $V = \{S, C\}$ $T = \{a, b, c\}$ $P = \{S \rightarrow abc \mid \varepsilon,$ $ab \rightarrow aabbC,$ $Cb \rightarrow bC,$ $Cc \rightarrow cc\ \}$	n=0	abc
	n=1	aabbCc
	n=2	aaabbCbCc
	n=3	aaabbCbcc
	n=4	aaabbbCcc
	n=5	aaabbbccc
c) Gramática G_3 $V = \{S, A, B, C, D, E\}$ $T = \{a, b, c\}$ $P = \{S \rightarrow aAB \mid aB,$ $A \rightarrow aAC,$ $A \rightarrow aC,$ $B \rightarrow Dc,$ $D \rightarrow b,$ $CD \rightarrow CE,$ $CE \rightarrow DE,$ $DE \rightarrow DC,$ $Cc \rightarrow Dcc\ \}$	n=0	aAB
	n=1	aaCB
	n=2	aaCDc
	n=3	aaCEc
	n=4	aaDEc
	n=5	aaDCc
	n=6	aaDDcc
	n=7	aabDcc
	n=8	aabbcc
d) Gramática G_4 $V = \{S, X\}$ $T = \{a, b\}$ $P = \{S \rightarrow bS \mid aX,$ $X \rightarrow bX \mid b\ \}$	n=0	bS
	n=1	bbS
	n=2	bbaX
	n=3	bbabX
	n=4	bbabb

Essa classe de gramática gera Linguagens Sensíveis ao Contexto e são reconhecidas por uma máquina de Turing com fita limitada. A gramática G_3, apresentada no Quadro 2, é um exemplo de gramática do Tipo 1. Também são apresentados oito níveis de produção no processo de derivação da palavra $aabbcc$.

Tipo 2 ou Gramáticas Livre de Contexto: trata-se da classe de gramáticas em que as regras de produção estão na forma $A \to \beta$ na qual $A \in V$ e $\beta \in (V \cup T)^*$. O termo "livre de contexto" significa que, durante a produção, os símbolos que antecedem e/ou sucedem A não são considerados, ou seja, A sempre produzirá β independente dos símbolos ao lado de A.

As linguagens geradas por essa classe de gramática são chamadas de Linguagens Livre de Contexto e são reconhecidas por um Autômato com Pilha. A gramática G_1, apresentada no Quadro 2, é um exemplo de Gramática Livre de Contexto ou Tipo 2.

Tipo 3 ou Gramáticas Regulares: trata-se da classe de gramáticas em que as regras de produção estão na forma $A \to Bw$ ou $A \to wB$, sendo $A \in V$ e $B \in T^*$.

Tal classe de gramática gera as chamadas Linguagens Regulares e são reconhecidas por um Autômato Finito. A gramática G_4, apresentada também no Quadro 2, é um exemplo de Gramática Regular. São apresentados ainda, quatro níveis de produção no processo de derivação da palavra $bbabb$.

2.2 *L-SYSTEM*

A interpretação biológica de Lindenmayer sobre as gramáticas de Chomsky, na tentativa de representar o crescimento celular de um organismo multicelular, resultou num formalismo que ficou conhecido como Sistemas de Lindenmayer ou, mais comumente, *L-systems*.

Tal como se dá no crescimento celular de um organismo, no qual várias divisões celulares ocorrem simultaneamente, o formalismo de Lindenmayer permite, por meio de uma gramática especial, a representação da reescrita paralela e simultânea, o que não ocorre nas gramáticas convencionais de Chomsky em que a reescrita ocorre sequencialmente.

O processo de reescrita paralela e simultânea é a característica que permite aos *L-systems* representar objetos que apresentem auto-similaridade, como fractais e plantas. A auto-similaridade é a propriedade de um objeto que indica que suas partes menores ou fragmentos apresentam a mesma forma do objeto inteiro, ou, possui forma estatisticamente similar a ele. Desse modo, essa propriedade é inerente à própria definição das gramáticas de Lindenmayer formalmente descritas na seqüência.

2.2.1 GRAMÁTICAS *L-SYSTEMS*

O formalismo dos *L-systems* é definido como uma gramática representada por uma quádrupla ordenada $G=\{V, T, P, S\}$, muito similar à definição de uma gramática convencional. É o que ocorre com os conjuntos V , T e o axioma S, com a observação de que T, neste caso, pode ser um conjunto vazio. Já no conjunto P, que contém as regras de produção representadas pelo par ordenado (α,β), α deverá ser um elemento de T^+. Deve-se notar, no entanto, que as regras de produção serão aplicadas em paralelo e simultaneamente. Todas as regras, para as quais houver correspondência para os símbolos variáveis na palavra a ser derivada no instante da produção, serão aplicadas ao mesmo tempo.

Na Figura 4, é possível notar a aplicação das regras de produção de forma simultânea e paralela em cada nível de produção da gramática G_5, apresentada no Quadro 3.

Em relação à forma das regras de produção, tal como ocorrem com as gramáticas de Chomsky, as gramáticas *L-systems* também poderão apresentar-se na forma **determinística** ou **não determinística**, além de poderem ser **estocásticas** quando houver pesos probabilísticos associados à suas regras de produção. Também poderá ser dita **parametrizada** a gramática

L-system que se fizer valer do uso de parâmetros junto aos símbolos variáveis, compondo funções. Isto, de certo modo, potencializa o controle e a influência das regras no ato da interpretação da cadeia resultante.

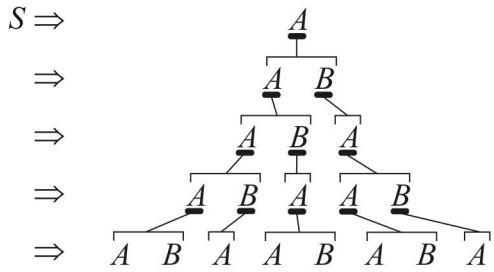


Figura 4 - Processo de derivação simultânea e paralela da gramática G_5 (*L-system*)

2.2.2 CLASSES DE GRAMÁTICAS *L-SYSTEMS*

As gramáticas *L-systems* foram organizadas em duas grandes classes: *L-systems* Livres de Contexto e L-systems Sensíveis ao Contexto. Os exemplos de gramáticas nessas classes são apresentados no Quadro 3.

O*L-systems* ou ***L-systems* Livres de Contexto:** nessas gramáticas as regras de produção atuam no processo de derivação sem considerar o contexto em que os símbolos deriváveis aparecem. As gramáticas determinísticas e livres de contexto são por vezes denotadas simplesmente por **DO*L-systems***. A gramática G_5, apresentada no Quadro 3, é um exemplo nessa classe.

I*L-systems* ou ***L-systems* Sensíveis ao Contexto:** trata-se das gramáticas em que as regras de produção atuam no processo de derivação, considerando-se o contexto em que os símbolos deriváveis aparecem, sejam pela esquerda, pela direita ou ambos os lados. As gramáticas sensíveis ao contexto também podem receber notação distinta quando se desejar apontar a posição de sensibilidade do contexto. Por exemplo, as gramáticas chamadas de **2*L-systems*** são gramáticas que estão sensíveis tanto à esquerda do contexto, como à direita. Já as gramáticas **1*L-systems*** estão sensíveis apenas a um lado do contexto, ou à esquerda ou à direita. Pode ocorrer, no entanto, que a

nomenclatura de uma gramática sensível ao contexto também aponte o tamanho da cadeia de caracteres considerada no contexto. Isso pode ser indicado simplesmente por ***(k,l)systems***, sendo k o tamanho da palavra à esquerda e l o tamanho da palavra à direita.

Quadro 3 - Exemplo de gramáticas *L-system* a) Livre de Contexto e b) Sensível ao Contexto

Definição	Nível	Produção
a) Gramática G_5 $V = \{S, A, B\}$ $T = \{\}$ $P = \{S \rightarrow A,$ $A \rightarrow AB,$ $B \rightarrow A\ \}$	n=0	A
	n=1	AB
	n=2	ABA
	n=3	ABAAB
	n=4	ABAABABA
b) Gramática G_6 $V = \{S, A, B\}$ $T = \{\}$ $P = \{S \rightarrow ABAABAA,$ $A > B \rightarrow B\ \}$	n=0	ABAABAA
	n=1	BBABBAA
	n=2	BBBBBAA

O exemplo de gramática nessa classe é a gramática G_6, apresentada no Quadro 3, juntamente com o processo de produção da palavra $BBBBBAA$ em dois níveis. Nessa gramática nota-se que, na regra $A > B \rightarrow B$, a sensibilidade do contexto é indicada por $A > B$, o que significa que essa regra somente produzirá B quando o símbolo A vier seguido de B.

2.2.3 RELAÇÃO ENTRE HIERARQUIA DE CHOMSKY E *L-SYSTEMS*

Embora haja intersecção entre as classes de linguagens presentes na hierarquia de Chomsky e as linguagens geradas pelos *L-systems*, nota-se que elas não são equivalentes conforme pode ser observado na relação de pertinência entre essas classes de linguagens na Figura 5, obtida com base em PRUSINKIEWICZ e LINDENMAYER (1990, p. 3).

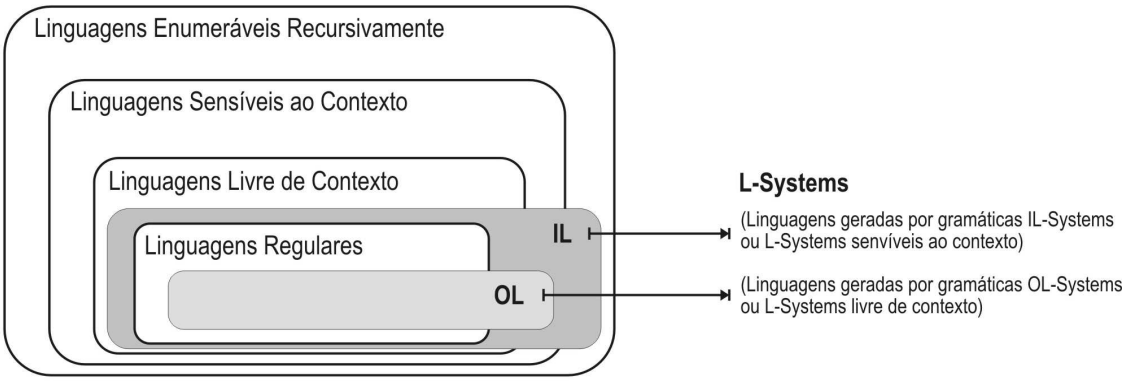


Figura 5 - Hierarquia de Chomsky e gramáticas *L-systems*

Pode ser observado que a classe de Linguagens OL está inteiramente contida na classe de Linguagens Sensíveis ao contexto, porém a classe de Linguagens OL faz intersecção entre as classes de Linguagens Regulares e Linguagens Livre de Contexto. Já a classe de Linguagens IL está inteiramente contida na classe de Linguagens Enumeráveis Recursivamente e faz intersecção com todas as outras classes.

2.2.4 INTERPRETAÇÃO GRÁFICA

A proposta de PRUSINKIEWICZ (1986a), que se tornou na abordagem clássica para a interpretação gráfica das gramáticas *L-systems*, assume que determinados símbolos existentes em uma cadeia produzida por uma gramática *L-system* deve atuar no construtor gráfico mediante analogias à geometria da tartaruga da linguagem LOGO (PAPERT, 1986).

A idéia básica dessa interpretação geométrica é dada por um cursor gráfico (que no caso da linguagem LOGO é a tartaruga) capaz de produzir traçados geométricos em coordenadas cartesianas mediante a alteração de estado da máquina que o controla.

O estado desse cursor é alterado mediante a troca de valores da tripla (x, y, α), sendo o par (x, y) a coordenada atual desse cursor a partir do qual será calculada, pelas relações trigonométricas do triângulo retângulo, a nova posição (x', y') dado o valor de deslocamento d (passos) desejado para

o cursor. Além disso, α é o ângulo que determinará a orientação incremental no qual se dará o deslocamento.

PRUSINKIEWICZ (1986a) também associou os símbolos $\{F, f, +, -, |, [,]\}$ aos comandos de alterações de estado do cursor conforme mostrados na seqüência.

F : desenha uma linha medindo d passos a partir da sua posição atual (x, y, α) até sua nova posição (x', y', α) dada por $x' = x + d.\cos(\alpha)$ e $y' = y + d.\text{sen}(\alpha)$;

f : desloca o cursor para frente d passos (como em F), porém sem desenhar nada;

$+$: gira o cursor à direita pela adição de δ ao ângulo α atual, alterando o estado do cursor para $(x, y, \alpha + \delta)$;

$-$: gira o cursor à esquerda pela subtração de δ ao ângulo α atual, alterando o estado do cursor para $(x, y, \alpha - \delta)$;

$|$: gira o cursor meia volta, adicionando 180º ao ângulo α atual, alterando o estado do cursor para $(x, y, \alpha + 180^\circ)$;

[:empilha o estado atual do cursor;

] :desempilha o último estado armazenado.

É importante observar que, as gramáticas analisadas neste trabalho utilizam essa mesma convenção simbólica idealizada por PRUSINKIEWICZ (1986a). Todas as gramáticas tratadas na metodologia proposta são constituídas por esses símbolos.

As gramáticas apresentadas no Quadro 4 são dadas para exemplificar o processo de interpretação gráfica de suas produções. As gramáticas G_7 e G_8 são utilizadas para obtenção dos exemplos de interpretações gráficas 2D, ambas gramáticas *L-systems* determinísticas e livre de contexto. G_7 é geradora do fractal conhecido como Ilha Quadrática de Koch (MANDELBROT, 1982) e G_8 é geradora de uma simples estrutura ramificada (PRUSINKIEWICZ; LINDENMAYER, 1990).

Na Figura 6, é apresentada a interpretação gráfica das cadeias produzidas pela gramática G_7.

Já na Figura 7, é apresentada a interpretação gráfica das cadeias produzidas pela gramática G_8, representando o processo de desenvolvimento de uma estrutura ramificada similar a um arbusto.

Também é possível a reconstrução de modelos tridimensionais baseados em *L-systems,* ampliando-se o significado dos símbolos já utilizados e adicionando-se outros associados à modificação de estado do cursor dentro do espaço 3D, contudo, agora, em uma notação vetorial unitária e perpendiculares entre si.

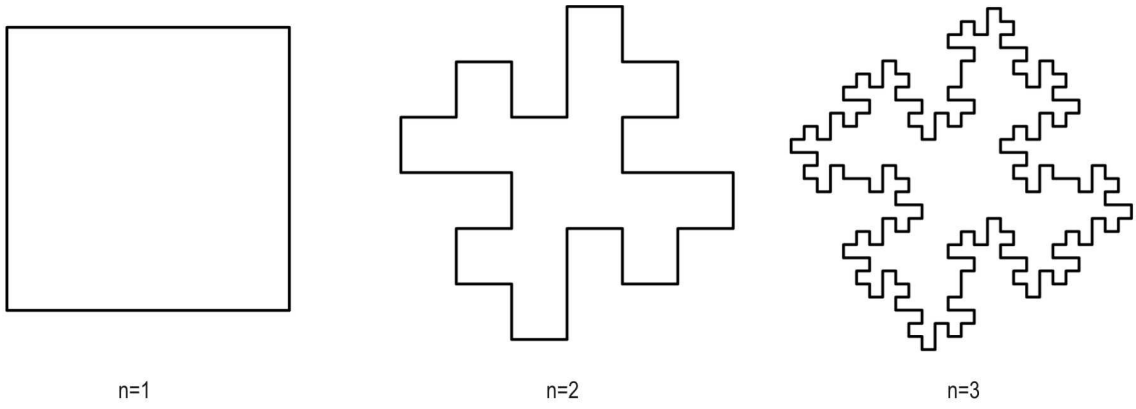


Figura 6 - Interpretação gráfica da gramática G_7

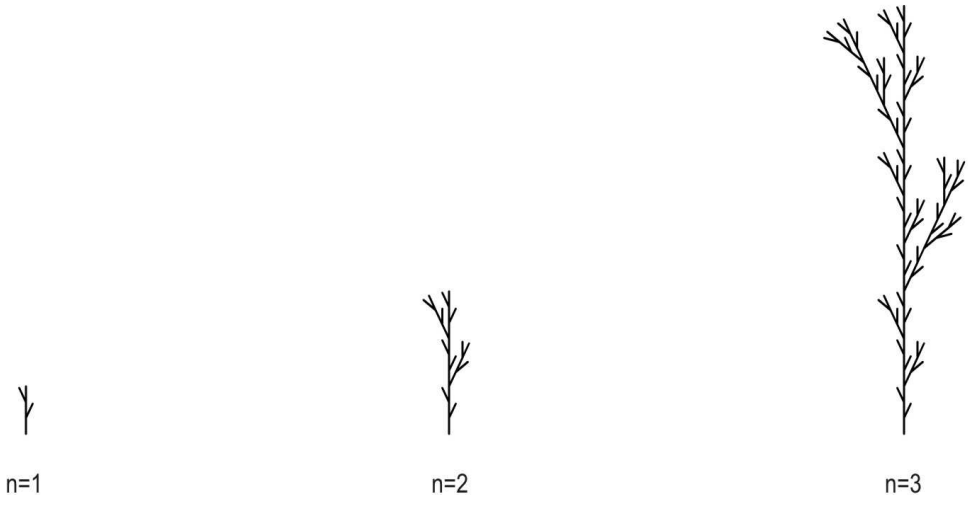


Figura 7 - Interpretação gráfica da gramática G_8

Quadro 4 - Exemplos de gramática *L-system* a) Ilha Quadrática de Koch, b) estrutura ramificada e c) Curva de Hilbert em 3D

a) Gramática G_7 $\alpha = 90^\circ$ $V = \{S, F\}$ $T = \{+, -\}$ $P = \{S \rightarrow F + F + F + F,$ $F \rightarrow F + F - F - F$ $F + F + F - F\}$	n=0	F+F+F+F
	n=1	F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F- FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-
	n=2	F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F+F-F- FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F- F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F+F-F- FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F- F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F- FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F- FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F- FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F- F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F- FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F- F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F
b) Gramática G_8 $\alpha = 25^\circ$ $V = \{S, F\}$ $T = \{+, -\}$ $P = \{S \rightarrow F,$ $F \rightarrow F[+F]F[-F]F\}$	n=0	F
	n=1	F[+F]F[-F]F
	n=2	F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[- F[+F]F[-F]F]F[+F]F[-F]F
	n=3	F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[- F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[- F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[- F]F]F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[- F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[- F]F[-F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[- F]F[-F[+F]F[-F]F]F[+F]F[-F]F]F[+F]F[- F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[- F]F]F[+F]F[-F]F
c) Gramática G_9 $V = \{S, A, B, C, D, F\}$ $T = \{+, -, /, \|, \&, \wedge\}$ $P = \{S \rightarrow A,$ $A \rightarrow$ *B-F+CFC+F- D &F ^ D-F+&& CFC+F+B//* $B \rightarrow$ *A&F ^ CFB ^ F ^ D^^-F-D ^|F ^ B|FC ^ F ^ A//* $C \rightarrow$ *|D^|F ^ B-F+C ^ F ^ A&&F ^ A &F ^ C+F+B ^ F ^ D//* $D \rightarrow$ *|CFB-F+B|F A& F ^ A&&FB-F+ B|FC// }*	n=0	A
	n=1	B-F+CFC+F-D&FD-F+&&CFC+F+B//
	n=2	A&FCFBFD-F-D|FB|FCFA//-F+|D|FB- F+CFA&&FA&FC+F+BFD//F|D|FB- F+CFA&&FA&FC+F+BFD//+F-|CFB- F+B|FA&FA&&FB-F+B|FC//&F|CFB- F+B|FA&FA&&FB-F+B|FC//-F+&&|D|FB- F+CFA&&FA&FC+F+BFD//F|D|FB- F+CFA&&FA&FC+F+BFD//+F+A&FCFBFD-F- D|FB|FCFA////

Nessa notação vetorial, $\vec{H}$ indica o movimento para frente ou para trás, $\vec{L}$ indica os movimentos para direita ou para esquerda e $\vec{U}$, indica os movimentos para cima ou para baixo. A rotação do cursor no espaço 3D é expressa a seguir, pela equação 1, na qual R é uma matriz de rotação 3x3 influenciada pelo ângulo α. A rotação dada pelo ângulo α sobre cada vetor está expressa nas equações 2, 3 e 4.

$$\left[\vec{H}' \quad \vec{L}' \quad \vec{U}'\right] = \left[\vec{H} \quad \vec{L} \quad \vec{U}\right] R \tag{1}$$

$$R_H(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\operatorname{sen}\alpha \\ 0 & \operatorname{sen}\alpha & \cos\alpha \end{bmatrix} \tag{2}$$

$$R_L(\alpha) = \begin{bmatrix} \cos\alpha & 0 & -\operatorname{sen}\alpha \\ 0 & 1 & 0 \\ \operatorname{sen}\alpha & 0 & \cos\alpha \end{bmatrix} \tag{3}$$

$$R_U(\alpha) = \begin{bmatrix} \cos\alpha & \operatorname{sen}\alpha & 0 \\ -\operatorname{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Alguns dos símbolos associados ao ambiente tridimensional são:

\+ : gira o cursor à esquerda pelo ângulo δ, usando a matriz de rotação $R_U(\delta)$;

\- : gira o cursor à direita pelo ângulo δ, usando a matriz de rotação $R_U(-\delta)$;

& : aponta o cursor para baixo pelo ângulo δ, usando a matriz de rotação $R_L(\delta)$;

^ : aponta o cursor para cima pelo ângulo δ, usando a matriz de rotação $R_L(-\delta)$;

\ : inclina o cursor à esquerda pelo ângulo δ, usando a matriz de rotação $R_H(\delta)$;

/ : inclina o cursor à direita pelo ângulo δ, usando a matriz de rotação $R_H(-\delta)$;

| : gira para trás usando a matriz de rotação $R_U(180^\circ)$;

A definição da gramática G_9, apresentada no Quadro 4, é um exemplo da utilização desses símbolos. A Figura 8 é resultante da interpretação gráfica das produções da gramática G_9 obtida por PRUSINKIEWICZ e LINDENMAYER (1990, p. 20). Trata-se de uma versão tridimensional do fractal conhecido como Curva de Hilbert (STEVENS, R. J.; LEHAR, A. F.; PERSTON, 1983). A cadeia de caracteres utilizada para a obtenção da Figura 8 não foi apresentada por ser demasiadamente grande.

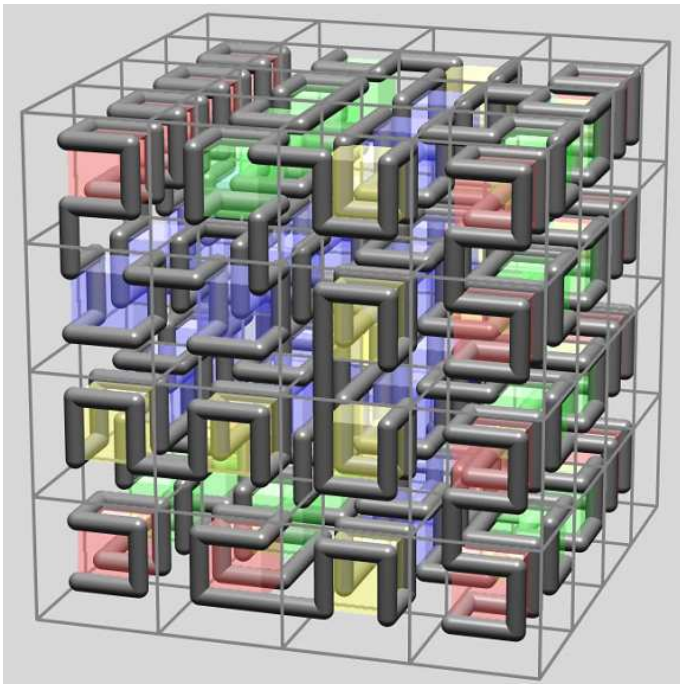


Figura 8 - Curva de Hilbert 3D obtida por PRUSINKIEWICZ e LINDENMAYER (1990, p. 20)

Ainda na Figura 8, a representação dos quadros coloridos foi estabelecida pela associação dos símbolos A, B, C e D com as cores vermelho, azul, verde e amarelo respectivamente, exemplificando outra forma

de se obter controles das propriedades dos objetos durante a interpretação gráfica.

2.3 INFERÊNCIA GRAMATICAL

A busca de formalismos que representem adequadamente modelos de objetos, processos ou fenômenos naturais tem atraído atenção de muitos pesquisadores em áreas como aprendizagem de máquina (HIGUERA, 2000), biologia computacional (SEARLS, 1993) (GRATE, 1994) (SAKAKIBARA, 1994), reconhecimento de padrões em documentos estruturados (SAIDI; TAYEB-BY, 1998) ou reconhecimento de fala (KASHYAP, 1979). Um dos formalismos utilizados para tentar alcançar tal representação tem sido a gramática. Nesse sentido, o objetivo de encontrar uma gramática para descrever apropriadamente um processo é a essência da chamada inferência gramatical, por vezes referenciada por indução gramatical, um subdomínio da inferência indutiva ou problema inverso. A inferência gramatical pode ser definida como um processo indutivo que pretende construir uma gramática geradora de uma linguagem desconhecida, utilizando-se de amostras finitas dessa linguagem (SAKAKIBARA, 1995). HIGUERA (2005) aponta esse problema como a busca ou identificação de uma função dado um conjunto de valores.

Comumente, duas abordagens são utilizadas ao se tratar de problemas de inferência gramatical (SAKAKIBARA, 1995):

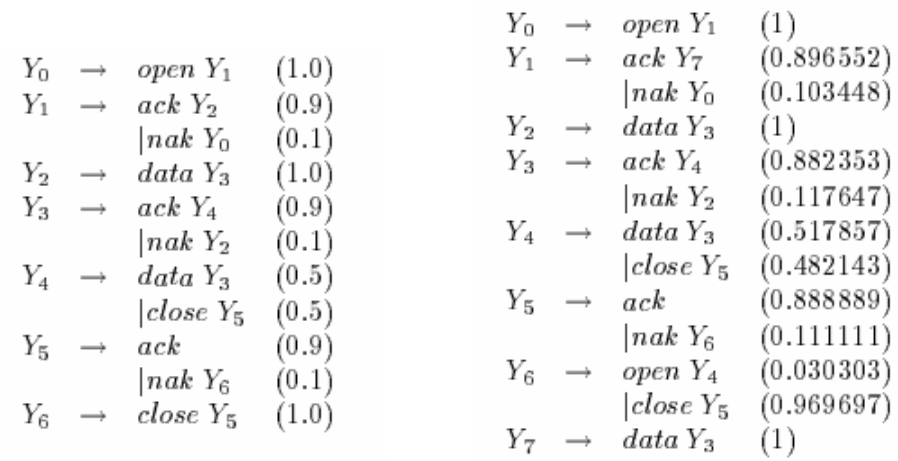
- pela enumeração de todas as possíveis gramáticas de uma determinada classe, testando-se uma a uma para verificar quais podem gerar as cadeias fornecidas como amostra;
- pela construção dessas gramáticas a partir dessas amostras, quando enumerar o espaço de soluções de gramáticas torna-se impraticável.

Para exemplo, é apresentado o trabalho de SCHWEHM e OST (1995), que propõe o uso de um ambiente de processamento paralelo com 1024 unidades processantes para execução de um pesado algoritmo genético capaz de inferir por construção uma gramática estocástica.

O algoritmo apresentado teria como ponto de partida um conjunto de palavras pertencentes à linguagem gerada por uma gramática desconhecida. O algoritmo inclui uma função objetiva para guiá-lo na procura de soluções viáveis, recompensando soluções menos complexas. Nos testes apresentados, foram utilizadas palavras geradas de uma gramática estocástica conhecida que constituía um simples protocolo de transferência de arquivo.

É apresentada, na Figura 9, a gramática original em (a) com sete regras de produção em comparação com a gramática inferida em (b), portando 8 regras de produção obtidas por Schwehm e Ost (1995, p. 7).

É observável que a gramática obtida nessa experiência é quase idêntica à gramática original, apenas sofrendo variações mínimas nas probabilidades das regras de produção, além da presença de uma regra de produção redundante ($Y_2 = Y_7$).



(*a*) *gramática original* (*b*) *gramática inferida*

Figura 9 - Exemplo de gramática inferida (SCHWEHM; OST, 1995, p. 7)

Nesse trabalho também é possível verificar que a convergência do algoritmo ao resultado ocorre apenas entre as gerações de número 60 e 80, sendo que, daí em diante, há apenas uma melhor aproximação dos pesos probabilísticos de cada regra encontrada.

2.4 PROBLEMA INVERSO DE LINDENMAYER

Um dos maiores problemas no formalismo dos *L-systems* está em determinar os modelos matemáticos que alcancem a melhor representação para um dado processo evolutivo. Como obter tais modelos ou gramáticas capazes de descrever satisfatoriamente, senão identicamente, tal evolução? Esse problema é conhecido como o **problema inverso de Lindenmayer**. Trata-se de uma particularidade da inferência gramatical, atuando no contexto do formalismo dos *L-systems* (PRUSINKIEWICZ; LINDENMAYER, 1990). Isso implica que no processo de busca de uma gramática devem ser consideradas as peculiaridades desse sistema como, por exemplo, a auto-similaridade presentes nos processos descritos pelos *L-systems* e seu desenvolvimento por reescrita paralela.

Muitas propostas têm sido apresentadas na busca de uma solução para o problema inverso de Lindenmayer. Vários desses trabalhos têm visualizado nos algoritmos genéticos grandes potenciais para a descoberta de soluções ou aproximações.

Um dos primeiros trabalhos mostrando a possibilidade de se descobrir regras de reescrita para resolver o problema inverso de Lindenmayer foi apresentado por KOZA (1993) baseado na programação genética, uma extensão de algoritmos genéticos. Nessa abordagem é gerada uma população inicial randômica de 4000 indivíduos compostos por conjuntos de símbolos terminais e conjuntos de funções primitivas candidatos à solução do problema. Essa população é baseada em uma divisão quadrática de 100x100 partes em torno do objeto alvo representante do segundo nível de produção da gramática da Ilha Quadrática de Koch (MANDELBROT, 1982) tomada como exemplo. Cada indivíduo é posto em execução e recebe uma pontuação de acordo com a maior proximidade do objeto original. Uma nova população é criada a partir da recombinação probabilística entre os indivíduos com as melhores pontuações formando uma nova geração. A cada geração o indivíduo que apresenta a melhor aproximação pode ser designado como solução da programação genética até aquele nível. Essa pode ser a solução exata ou apenas uma aproximação. Os limites para o fim da execução do algoritmo são

dados quando a solução atinge 100% de aceitação ou é alcançado o limite de 51 gerações, devido ao tempo computacional exigido e limites de memória. É observável que os resultados apresentados, considerando-se um objeto gerado por apenas duas derivações de um fractal extremamente simples, foram convergindo, atingindo a melhor aproximação somente na geração de número 50, revelando uma metodologia bastante custosa computacionalmente. Nesse trabalho também ficam evidenciadas algumas restrições como: necessidade do conhecimento prévio do axioma e do ângulo de ramificação obtido da representação gráfica do objeto. Além disso, a gramática deve ser determinística e livre de contexto com apenas uma única regra de produção. Na regra encontrada nesse trabalho também são feitos alguns ajustes manuais para torná-la idêntica à regra geradora original.

Regra original:

$F \rightarrow F\text{-}F\text{+}F\text{+}FF\text{-}F\text{-}F\text{+}F$

Regra encontrada:

$F \rightarrow F\text{-}F\text{+}F\text{--}\text{+}\text{+}\text{+}FF\text{-}F[\text{-}F]\text{-}F\text{+}F$

A regra original utilizada é igual à regra da gramática G_7 já apresentada no Quadro 4, exceto pelo fato dos símbolos estarem dispostos na ordem inversa de ocorrência. Isso não traz nenhuma implicação na reconstrução do fractal em questão, pois a interpretação gráfica produz a mesma figura.

A problemática de se encontrar a formalidade dos *L-system* se estende em outras abordagens como mostra o trabalho de SHLYAKHTER et al. (2001) que apresenta uma metodologia para reconstrução de modelos tridimensionais de árvores folhadas a partir de imagens reais. A idéia é primeiramente reconstruir o tronco e algumas ramificações de primeiro nível por informações obtidas de imagens tiradas em torno de uma árvore real. Em seguida, é utilizada uma gramática *L-system* para reconstrução das ramificações subseqüentes. Nessa gramática, o axioma é extraído da própria forma do tronco e galhos principais. As ramificações são obtidas pela derivação

dessa gramática na qual, em cada ramo gerado, é colocado um broto, indicando a possibilidade da continuidade do crescimento naquele ponto. Todo ramo que sair do envoltório visual, construído para obter o esqueleto do tronco, é podado e, no local, é colocado um novo broto. Os resultados apontam uma abordagem interessante para a construção de modelos realísticos de árvores folhadas, no entanto, não são apresentadas as regras de produção utilizadas, nem tampouco a formalidade da gramática completa.

Já no trabalho de RUDOLPH e ALBER (2002) é proposto o uso de um algoritmo evolucionário para resolver o problema inverso no projeto de torres de transmissão. A proposta é baseada na definição de uma tabela de formas geométricas primitivas associadas às regras de produção de uma gramática base, capaz de aceitar parâmetros para controle de características dessas primitivas, tais como raio, altura, largura, inclinação, etc. Além disso, o modo como o algoritmo foi construído permite que a estrutura da gramática seja modificada durante o processo gerativo (mutação de regras, ordem de ocorrência, exclusão e adição de regras, além de outras características) de forma a auxiliar a convergência do processo. A população de gramáticas iniciais é criada randomicamente. A avaliação de cada indivíduo obtido pelo algoritmo é realizada por uma função-objetiva. Essa função determina, por comparações de similaridades no espaço geométrico 3D, o quanto o indivíduo analisado se aproxima do(s) modelo(s) fornecido(s) como genótipo pretendido.

Na Figura 10 é apresentado o objeto pretendido, caracterizando o genótipo que o algoritmo proposto deveria convergir.

Já na Figura 11, é apresentado um gráfico de convergência do algoritmo durante a procura do objeto pretendido. Também é possível observar a relação de similaridade (distância) e o número de gerações criadas para a convergência do algoritmo. Abaixo do gráfico são apresentados alguns indivíduos obtidos durante o processo de convergência. A melhor aproximação foi obtida somente na geração de número 148.

Apesar dos resultados mostrarem uma interessante abordagem sobre o problema inverso, RUDOLPH e ALBER (2002) não apresentam a gramática obtida pelo algoritmo evolucionário para a reconstrução do modelo apresentado.

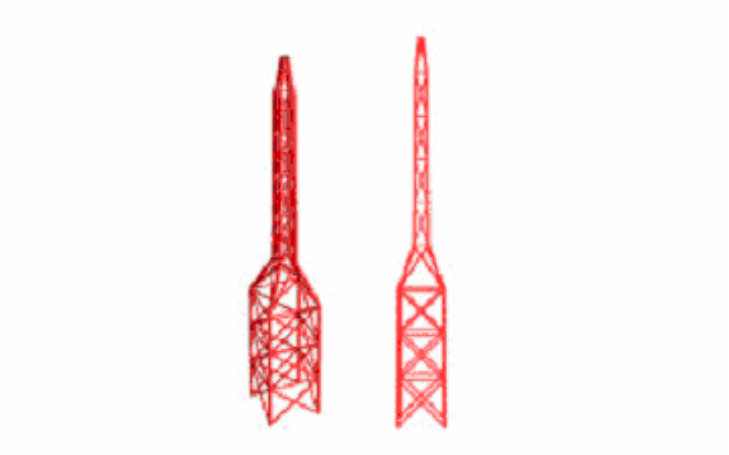


Figura 10 - Exemplo de genótipo pretendido (RUDOLPH; ALBER, 2002, p. 21)

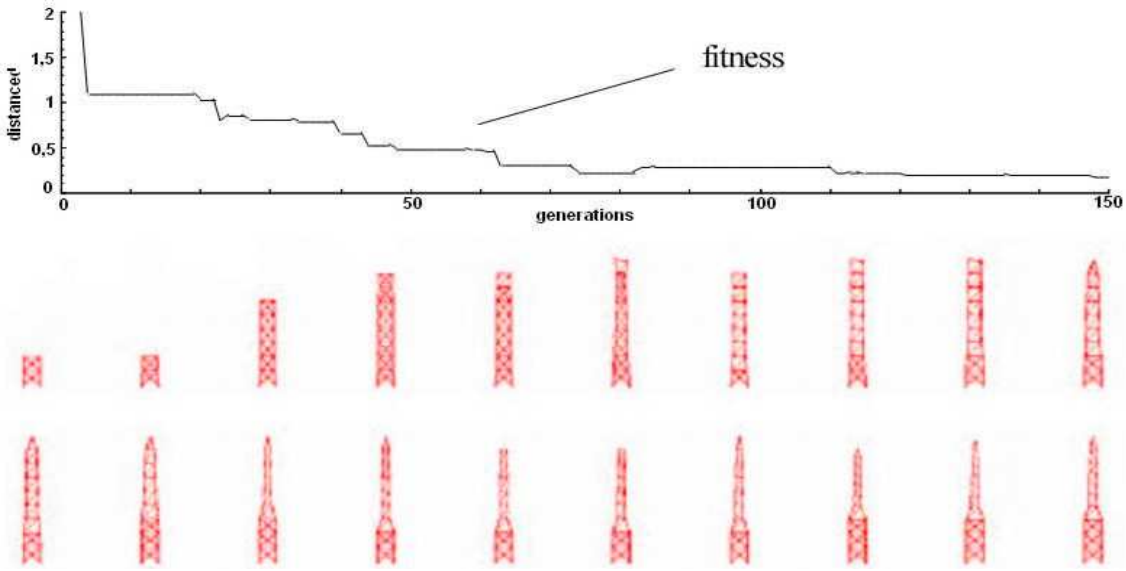


Figura 11 - Convergência do algoritmo proposto (RUDOLPH; ALBER, 2002, p. 21)

Outra abordagem que também se utiliza de algoritmos genéticos é o trabalho de Costa e Landry (2005), que propõe um método para descrever formalmente fractais em forma de árvores e reconstruir esses modelos. A idéia é gerar uma família de gramáticas sensíveis ao contexto a partir de combinações de operadores de crescimento (formas básicas predefinidas) com um axioma dado pela forma de uma árvore jovem. Dada uma lista de imagens representantes de um determinado espaço temporal do crescimento de uma árvore, o algoritmo deveria procurar uma gramática dessa

família cuja interpretação geométrica da derivação melhor se aproximasse dessa lista de figuras. Para os testes, foram geradas imagens a partir de gramáticas formadas com axiomas conhecidos contendo, como única regra, um dos operadores de crescimento. O algoritmo deveria obter não somente a mesma figura pela interpretação gráfica da cadeia gerada pela gramática encontrada, mas também essa gramática deveria ser igual à gramática original usada para reconstruir as figuras iniciais usadas como amostra. O parâmetro de convergência ou *fitness* usado no algoritmo genético teria que ser maior que zero caso contrário, seria necessário alto poder de processamento, memória e longo período de tempo para obter uma melhor convergência. Mesmo assim, o algoritmo alcançou apenas resultados similares às amostras na maioria dos casos. Não foram apresentadas as gramáticas encontradas nem qualquer regra de produção resultante do processo.

Diante dos trabalhos apresentados, fica claro compreender a complexidade do problema inverso de Lindenmayer, mesmo quando são tratados problemas aparentemente mais simples, como por exemplo, encontrar a regra geradora de um simples fractal ou arbusto ramificado.

Os trabalhos apresentados, mesmo fazendo uso de algoritmos genéticos, atuando no espaço de soluções por métodos evolucionários, tentaram convergir à solução, sem, contudo, alcançar a exatidão.

É possível notar que a questão ainda permanece aberta: dada uma seqüência de caracteres representante de um objeto, em um determinado nível de desenvolvimento é possível identificar a(s) regra(s) que o descreve(m) ou mesmo determinar uma gramática que o descreva? É possível uma solução para o Problema Inverso de Lindenmayer, ou pelo menos parte dele, tendo como fundamentos as peculiaridades desses sistemas sem haver a necessidade de explorar a dimensão de possíveis soluções?

3 OBTENÇÃO DE REGRAS PARA GRAMÁTICAS *L-SYSTEMS* LIVRES DE CONTEXTO E DETERMINÍSTICAS

Para o desenvolvimento deste trabalho, foi necessário, primeiramente, determinar os limites de exploração no espaço das classes de gramáticas *L-systems*, e somente então, fundamentar a metodologia nesses limites. Sendo assim, tais considerações são apresentadas a seguir nas subseções 3.1 e 3.2.

3.1 DELIMITAÇÃO DO PROBLEMA

O Problema Inverso de Lindenmayer é visto como um problema de alta complexidade e não é tarefa simples resolvê-lo. Este trabalho explorou as propriedades e relações das gramáticas *L-systems* Livres de Contexto e Determinísticas que possuem a restrição de terem o axioma formado por um único símbolo, sendo constituídas por uma única regra de produção.

Também é importante considerar que, as cadeias de caracteres utilizadas nesse trabalho foram obtidas a partir de gramáticas conhecidas conforme as restrições anteriores, porém, foram tratadas como sendo de origem desconhecida a fim de permitir a validação da proposta. A intenção foi encontrar uma metodologia capaz de encontrar uma regra *L-system*, utilizando, para isso, somente uma dessas cadeias de caracteres independente do nível de produção em que tal cadeia tenha sido originada.

O processo de reescrita paralela e simultânea que a cadeia sofreu ao longo das produções é o único fator conhecido. Todas as demais informações para solução do problema são obtidas a partir da análise da cadeia. A observação das propriedades na formação dessas cadeias, durante a análise dos vários níveis de produção de gramáticas conhecidas, é que habilitaram a formulação de hipóteses sobre a lei de regressão dessas cadeias.

3.2 METODOLOGIA PROPOSTA

As ações adotadas inicialmente consistiram na análise e comparação de gramáticas *L-systems* pertencentes às restrições indicadas anteriormente. Dessa análise foi possível identificar relações e propriedades que serão descritas a seguir, nas próximas subseções, considerando-se, para exemplo, a gramática G_8 já apresentada no Quadro 4 da subseção 2.2.4.

3.2.1 CONSIDERAÇÕES SOBRE A REVERSÃO DA CADEIA

A seguir, no esquema de produção da gramática G_8 representado na Figura 12, é possível notar que, a partir do primeiro nível de produção, todos os demais níveis são compostos exatamente pelas mesmas partes. Essas partes são resultantes do formato da regra de produção desta gramática, determinado pela presença dos símbolos terminais atuando como separadores dos pontos de evolução. Os balões indicam, exatamente, estes pontos na cadeia em produção. É notável que, mesmo após a reescrita exaustiva das produções, estes pontos sempre estarão separados pelos mesmos símbolos terminais. Deve-se notar que o nível zero não é listado na representação, pois indica apenas o axioma e não é resultado de uma produção. Além disso, devido a gramática ser constituída de uma única regra, o primeiro nível de produção será sempre idêntico a essa regra.

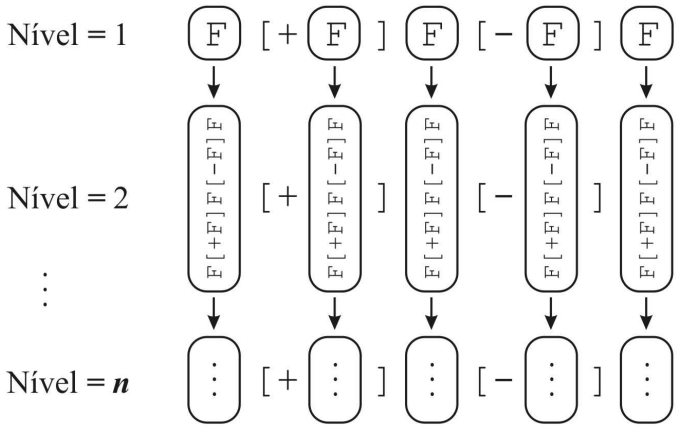


Figura 12 - Esquema de produção da gramática G_8

Nessa figura também é possível observar que em cada nível de produção foi mantida a mesma estrutura da regra definida pela gramática. Na definição da regra na gramática G_8 existem “cinco $'F's$” separados por alguns símbolos terminais e, no esquema dessa figura, também existem “cinco partes” indicadas pelos balões e separadas, exatamente, pelos mesmos símbolos terminais presentes na regra independente do nível de produção. Pode ser visto com muita clareza que os $'F's$ geradores presentes na regra de produção são exatamente os pontos onde se dará o crescimento da cadeia pelo processo de reescrita. Pelo fato do nível um representar a própria regra de produção, todos os demais níveis assumirão o mesmo formato estrutural.

Pode-se compreender que em qualquer nível de produção dessa gramática sempre haverá cinco partes separadas exatamente pelos mesmos símbolos terminais presentes na regra de produção em $'F'$. Dessa forma, fica claro, que se o formato estrutural de uma cadeia em um determinado nível de produção for conhecido, será possível, então, inferir o formato estrutural da regra de produção e vice-versa. Reconhecer o formato estrutural de uma cadeia significa saber quantas e quais partes são resultantes dos $'F's$ geradores.

Sendo assim, a idéia proposta é reverter o crescimento da cadeia promovido pelo processo de reescrita, de uma só vez e em todos estes pontos de evolução indicados pelos balões, tal como mostra o esquema da Figura 13.

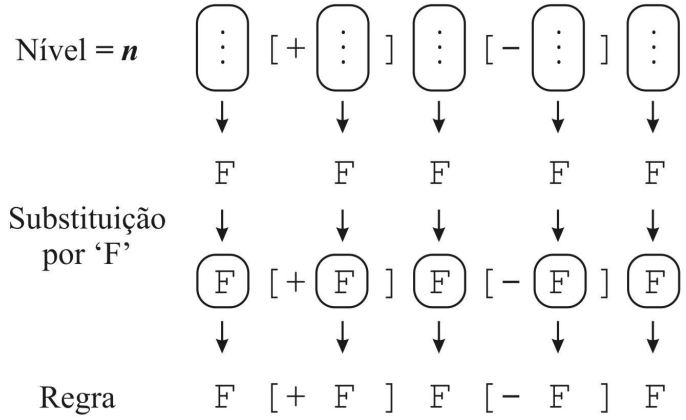


Figura 13 - Esquema de reversão para a cadeia em análise

Contudo, no processo de busca da regra geradora, o grande problema está em identificar os pontos de evolução na cadeia para que se possa efetuar o processo de regressão. Convém notar que não se sabe onde começa ou termina cada um desses pontos e nem quais são os símbolos que os estão separando. Além disso, para assegurar que o processo de reversão resulte em uma regra potencialmente capaz de reproduzir a cadeia analisada, é necessário considerar alguns fatores primordiais:

i. deverá ser encontrada qual quantidade de $'F's$, em uma regra *L-system*, poderia matematicamente resultar na cadeia analisada;
ii. deverá ser verificado se a reversão promovida na cadeia analisada produz a mesma quantidade de $'F's$ encontrada em (i);
iii. deverá ser verificado se a cadeia obtida na reversão é capaz de produzir a mesma quantidade de símbolos terminais presentes na cadeia analisada.

3.2.2 IDENTIFICANDO A ORDEM DE CRESCIMENTO DA CADEIA

Feitas as considerações anteriores, para cada nível de produção da gramática G_8, obteve-se o número de ocorrências do símbolo $'F'$ existentes na cadeia gerada pelo processo de derivação, já que esse é o símbolo variável utilizado na maioria das gramáticas que estão inclusas nas restrições apresentadas na subseção 3.1. Esse número de ocorrência será representado deste ponto em diante por F_t. A seguir, é possível visualizar, no Quadro 5, os valores de F_t encontrados em cada nível de produção da gramática G_8.

Quadro 5 - Contagem de *'F's* nas produções da gramática G_8

Nível de Produção	F_t
n = 0	1
n = 1	5
n = 2	25
n = 3	125

Essa contagem possibilitou observar a seguinte relação: uma vez que a regra de produção fará substituições somente sobre os símbolos $'F's$ presentes na cadeia a ser derivada, a contagem desses símbolos na cadeia resultante permitiu verificar que a ordem de crescimento dessa produção ocorreu de forma exponencial. Para cada símbolo $'F'$ existente na cadeia de origem, são acrescidos mais 5 símbolos $'F's$ por nível de produção, conforme as regras definidas na gramática G_8.

Assim, é possível representar F_t como uma relação exponencial entre a quantidade (F_q) de símbolos $'F's$ presentes na regra de produção e o nível de produção (n). Essa relação é expressa pela equação 5.

$$F_t = F_q{}^n \tag{5}$$

No Quadro 6 pode ser verificada a validade dessa relação para a gramática G_8 sendo $F_q = 5$, aplicando-se a equação 5 em cada nível de produção referenciado anteriormente.

Quadro 6 - Relação de crescimento da gramática *G_8*

Nível de Produção	F_t	$F_q{}^n$
n = 0	1	$5^0 = 1$
n = 1	5	$5^1 = 5$
n = 2	25	$5^2 = 25$
n = 3	125	$5^3 = 125$

3.2.3 FORMATO ESTRUTURAL DA REGRA PROCURADA

Uma vez entendendo-se que a ordem de crescimento estampada em qualquer nível de produção é de ordem exponencial e que essa ordem representa o número de $'F's$ existentes na regra de produção, como já mostrado na equação 5, então encontrar esse número significa reconhecer o formato estrutural da regra procurada, isto é, significa saber quantas partes em $'F'$ a regra desconhecida é composta. Procura-se então, um número (F_q) que efetuado seu produto por ele mesmo n vezes resulte na quantidade de $'F's$

(F_t) encontrada na cadeia de caracteres de um determinado nível de produção. Nesse processo, como apenas F_t é conhecido pela contagem de $'F's$, então é proposto que seja feita uma busca iterativa em F_q para encontrar valores para o expoente n capazes de produzir F_t. A iteração em F_q tem o seu fim no maior valor possível de uma exponenciação capaz de produzir F_t, o que equivale à parte inteira de sua própria raiz quadrada (excluindo-se qualquer base de expoente 1).

Assim, considerando-se a relação da equação 5, pode ser estabelecida a relação inversa por meio de uma função logarítmica, mostrada a seguir, na equação 6, para determinar os valores de n que satisfaçam a igualdade.

$$\log_{F_q} F_t = \frac{\ln F_t}{\ln F_q} = n \tag{6}$$

Uma vez que a iteração em F_q obtenha um valor inteiro para n, é possível reconhecer o suposto formato estrutural da regra de produção da cadeia analisada, ou seja, é possível saber quantos $'F's$ estarão presentes na provável regra de produção, e qual o nível em que a cadeia analisada pode ter sido produzida. Nota-se que F_t não pode ser resultante de um expoente fracionário. Isso assegura a primeira consideração apresentada em (i) na subseção 3.2.1. Nesse ponto, com os valores de F_q e n atuais, já é possível executar a operação de reversão, que será descrita na próxima subseção.

É importante notar, ainda, que poderá haver outros valores para F_q e n capazes de produzir o valor de F_t. É o caso de uma cadeia que apresente 64 $'F's$ ($F_t = 64$). Nesse caso, o valor de F_t poderá ser resultado das potências 2^6, 4^3, 8^2 e 64^1, sendo cada uma delas igualmente candidatas à solução do formato estrutural de uma cadeia com essas condições. Como os valores de F_q e n serão obtidos de acordo com os passos da iteração, quando o processo de reversão obtiver sucesso, as próximas iterações não serão

executadas, condicionando o algoritmo interromper o processo assim que a primeira solução for encontrada.

O código parcial contendo as considerações feitas até aqui, necessárias para obtenção dos valores de F_q e n, é apresentado no Algoritmo 1. Na linha 1, a função ***countSimbol()*** é responsável por obter a contagem dos símbolos $'F's$ existentes na cadeia fornecida. Também é importante considerar que, no início do processo de busca, uma cadeia que contenha zero ou apenas um $'F'$ não pode representar uma estrutura em desenvolvimento, sendo assim, a busca não tem significado e deve ser encerrada, retornando como solução a mesma cadeia fornecida ao algoritmo. Isso é realizado pelo desvio condicional da linha 2. O retorno da mesma cadeia entrada também ocorre quando $F_t > 1$ e mesmo assim não houver solução encontrada pelo processo de reversão, o que representará uma cadeia resultante do primeiro nível de produção sendo $F_t = F_q{}^1$. Na linha 3, o laço de repetição realiza a iteração em F_q, iniciando em 2 e encerrando até o valor inteiro da própria raiz quadrada de F_t, determinando o limite da iteração. Já na linha 4, é obtido o valor de n capaz de indicar uma exponenciação de F_q que produza F_t. Contudo, n deve ser um número inteiro para representar um nível de produção. Essa verificação é realizada pelo desvio condicional da linha 5. Se um valor inteiro for encontrado, é executado então o processo de reversão e validação; caso não haja êxito, segue-se para a próxima iteração.

No Quadro 7 são apresentados os valores de F_t, F_q e n obtidos pelo Algoritmo 1 executado sem interrupções, considerando como entrada a cadeia w, resultante do terceiro nível de produção da gramática G_8. Neste quadro, está evidenciado pelo sombreamento o quarto nível de iteração, indicando os valores de F_t e F_q que resultaram um valor inteiro para n. Esses valores apontam para uma cadeia que foi gerada, possivelmente, por uma regra contendo cinco $'F's$ ($F_q = 5$) ao terceiro nível ($n = 3$) de produção.

Algoritmo 1 - Procura valores para F_q e n

```
Entrada
   w: cadeia de caracteres
Saída
   regra: solução encontrada
Início
1   Ft←countSimbol('F', w)
2   se Ft>1 faça
3      para Fq←2 até int(sqrt(Ft)) faça
4           n←ln(Ft)/ln(Fq)
5           se isInteger(n) então
                        ⋮
6              (processo de reversão e obtenção da regra)
7              (validação da regra  )
8              (retornar regra      )
                        ⋮
9           fim do se
10     fim do para
11  fim do se
12  regra←w
13  retornar regra
Fim
```

Quadro 7 - Valores obtidos pelo Algoritmo 1 sobre a cadeia *w*

Iteração nº	Ft	Fq	$n = \log_{F_q} F_t$
1	125	2	6,96578
2	125	3	4,39492
3	125	4	3,48289
4	125	5	3
5	125	6	2,69473
6	125	7	2,48126
7	125	8	2,32193
8	125	9	2,19746
9	125	10	2,09691
10	125	11	2,01356

3.2.4 PROCESSO DE REVERSÃO DA CADEIA

Uma vez obtidos os valores de F_q e n, já é possível identificar os pontos de evolução na cadeia w. Como o valor de F_q indica que existem 5 $'F's$ presentes na regra procurada, e que cada $'F'$ foi responsável pela produção de uma parte da cadeia, então haverá 5 partes iguais nessa cadeia e

cada uma dessas partes conterá 25 caracteres 'F', pois 125 'F's divididos em 5 partes resultam em 5 grupos de 25 'F's.

Segue-se então para a identificação dessas partes. Aqui optou-se por identificar o primeiro agrupamento, da esquerda para a direita, presente na cadeia w. Percorre-se a cadeia a partir do seu início e guarda-se uma subcadeia s, partindo da ocorrência desde o primeiro 'F' encontrado nessa cadeia até o $k-ésimo$ 'F', sendo $k = F_t / F_q$.

Em seguida, a cadeia w deverá ser pesquisada a fim de se encontrar todas as ocorrências de s. Para promover a reversão do crescimento provocado pela regra de produção, cada ocorrência de s deverá ser substituída por um único 'F', de uma só vez. O efeito da substituição das ocorrências de s causará a regressão da cadeia principal diretamente à regra de produção procurada para a gramática G_8. Nesse processo, a contagem do número de substituições efetuadas também deverá ser feita. Essa contagem servirá para verificar se o número de substituições ocorridas é igual ao número de partes que a cadeia indica ter.

No Quadro 8, aparecem indicadas pelo sombreamento claro e escuro as ocorrências de s em w. Neste quadro, também pode ser notada a cadeia resultante do processo de reversão ainda com os pontos revertidos indicados pelo sombreamento, além da contagem de substituições efetuadas.

Quadro 8 - Identificação dos pontos de evolução e reversão da cadeia

Cadeia w com ocorrências de s sombreadas:	
F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F	
Cadeia resultante da reversão:	**Contagem das substituições:**
F[+F]F[-F]F	5 substituições

Após esse processo, deve ser verificado se a contagem das substituições efetuadas é igual ao valor indicado por F_q. Essa verificação assegura a consideração apresentada em (ii) na subseção 3.2.1. Para o caso

da cadeia w, foram efetuadas 5 substituições, valor igual ao encontrado em $F_q = 5$.

As considerações sobre o processo de reversão composto pela identificação da cadeia s e pela operação de substituição de s em w, obtendo-se a contagem das substituições efetuadas, foram sintetizadas nas funções ***slice()*** e ***replace()***, respectivamente. Ambas as funções estão presentes nas linhas 6 e 7 do código parcial do Algoritmo 2.

Algoritmo 2 - Processo de Reversão

```
Entrada
   w: cadeia de caracteres
Saída
   regra: solução encontrada
Início
1   Ft←countSimbol('F', w)
2   se Ft>1 faça
3      para Fq←2 até int(sqrt(Ft)) faça
4           n←ln(Ft)/ln(Fq)
5           se isInteger(n) então
6              s← slice(w, Ft, Fq)
7              se replace(regra,s,w)=Fq então
                          ⋮
8                 (validação da regra  )
9                 (retornar regra      )
                          ⋮
10             fim do se
11          fim do se
12     fim do para
13  fim do se
14  regra←w
15  retornar regra
Fim
```

A função ***slice()*** recebe como parâmetros os valores de F_t e F_q, além da cadeia w, retornando em s a subcadeia correspondente ao primeiro ponto de evolução das produções. Já a função ***replace()*** recebe como parâmetros os valores de s e w. O parâmetro $regra$ servirá para armazenar a cadeia resultante do processo de reversão. Ainda na linha 7, é feita a verificação se o processo de reversão resultou no mesmo valor indicado por F_q. Caso esses valores sejam diferentes, segue-se para a próxima iteração.

3.2.5 LIMPEZA DE SÍMBOLOS TERMINAIS ACUMULADOS

Algumas regras de produção podem promover o acúmulo de símbolos terminais na cadeia. É o caso, por exemplo, de uma regra como $F+F-$. Na Figura 14, são apresentados 3 níveis de produção dessa regra indicando, em cada nível, as partes produzidas pelos $'F's$ do nível anterior. Esta figura também indica a subcadeia retornada pela função ***slice()*** aplicada à produção do terceiro nível seguido da substituição dessa subcadeia pela função ***replace()***. Ao final, pode ser notado, na cadeia resultante, o acúmulo de símbolos terminais indicados pelos balões.

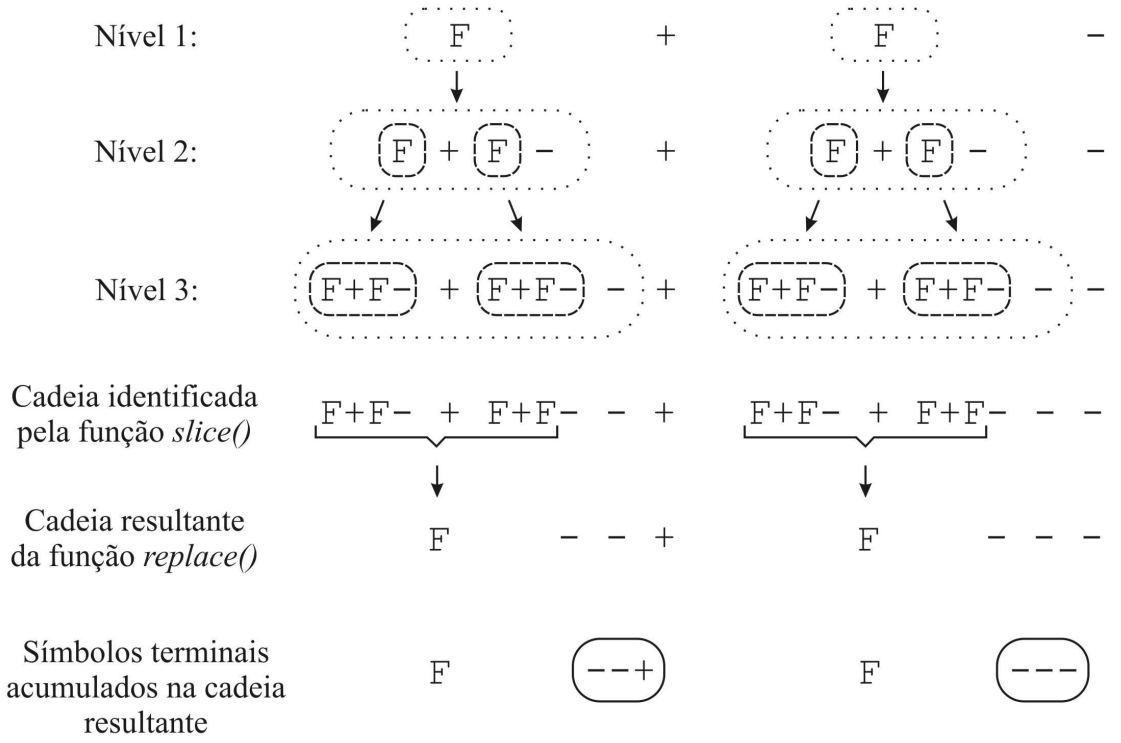


Figura 14 - Acúmulo de caracteres terminais

Para obtenção da regra, é necessário que esses símbolos terminais sejam eliminados na mesma proporção em que foram acumulados durante o processo de reescrita. Isso é feito identificando-se, na cadeia, quantos caracteres foram acumulados por nível de produção antes da primeira ocorrência de $'F'$. Dessa forma, divide-se a quantidade de símbolos terminais existentes antes da primeira ocorrência de $'F'$ por n. O quociente obtido indica

quantos símbolos terminais deverão ser retirados antes de cada 'F' por nível de produção. Nota-se que o primeiro nível deve ser excluído do cálculo pois representa a própria regra de produção, então, multiplica-se o quociente obtido por $n-1$. O produto representará a quantidade total de símbolos terminais que deverão ser excluídos antes de todas as ocorrências de 'F'. Esse procedimento também deve ser executado para os símbolos existentes depois de cada ocorrência de 'F' na sua devida proporção. Essas relações são apresentadas nas equações 7 e 8.

$$R_e = (T_e / n) * (n-1) \quad (7)$$

$$R_d = (T_d / n) * (n-1) \quad (8)$$

Na equação 7, o valor de T_e indica quantos símbolos estão presentes à esquerda da primeira ocorrência de 'F' na cadeia resultante da função ***replace()***. Já o valor de R_e representa a quantidade total de símbolos terminais que devem ser removidos à esquerda de todas as ocorrências de 'F' dessa cadeia. Nessa equação é usado o valor atual de n obtido pela iteração em F_q. A mesma relação está apresentada na equação 8, contudo, T_d referencia os símbolos terminais à direita da última ocorrência de 'F', e R_d referencia a quantidade de símbolos terminais que devem ser removidos à direita de todas as ocorrências de 'F'.

No Quadro 9, é possível acompanhar a aplicação das equações 6 e 7 sobre a cadeia retornada pela função ***replace()*** apresentada no esquema da Figura 14. Convém observar que a cadeia original foi obtida no terceiro nível de produção ($n=3$).

Os valores para R_e e R_d obtidos no Quadro 9 indicam respectivamente, que não deverá ser removido qualquer símbolo à esquerda das ocorrências de 'F', ao passo que à direita dessas ocorrências, deverão ser removidos 2 símbolos terminais de uma só vez.

Os procedimentos para limpeza de símbolos terminais foram organizados em uma função chamada ***terminalClear()*** que será incluída na listagem completa do código do Algoritmo 3, mais adiante na próxima

subseção. Essa função recebe como parâmetros a cadeia *regra* resultante do processo anterior e o valor de n. Não é feita nenhuma alteração em *regra* caso seja verificado que a cadeia não apresenta acúmulo de símbolos terminais.

Quadro 9 - Limpeza de Símbolos Terminais

<table>
<tr><td colspan="2">Cadeia retornada pela função replace():
F--+F---</td></tr>
<tr><td>Total de símbolos antes do primeiro $'F'$
$T_e = 0$</td><td>Total de símbolos depois do último $'F'$
$T_d = 3$</td></tr>
<tr><td>Aplicação da equação 7:
$R_e = (T_e / n) * (n-1)$
$R_e = (0/3) * (3-1)$
$R_e = 0 * 2$
$R_e = 0$</td><td>Aplicação da equação 8:
$R_d = (T_d / n) * (n-1)$
$R_d = (3/3) * (3-1)$
$R_d = 1 * 2$
$R_d = 2$</td></tr>
</table>

3.2.6 PROJEÇÃO DA QUANTIDADE DE SÍMBOLOS TERMINAIS

Finalmente, após a obtenção da cadeia resultante do processo de reversão armazenado em *regra*, já com a limpeza de símbolos terminais efetuada, a última consideração feita na subseção 3.2.1 deve ser analisada.

Esta análise deve verificar se a cadeia obtida pelo processo de reversão é capaz de produzir a mesma quantidade de símbolos terminais existentes em w sem que haja necessidade de efetuar as produções até o nível indicado por n. Como no processo de reescrita, os símbolos terminais vão sendo acumulados nível a nível, partindo da própria regra, bastaria saber quantos símbolos terminais estariam presentes em cada nível, e somá-los.

Também deve ser notado que para cada $'F'$ existente na cadeia derivada é inserida a mesma quantidade de símbolos terminais presentes na regra de produção, além dos símbolos já existentes resultantes da última derivação. Dessa forma, a quantidade de símbolos terminais existentes em um determinado nível de produção poderá ser encontrada pela equação 9.

$$S_r = \sum_{i=0}^{n-1} F_q^i * T \quad (9)$$

Nessa equação, F_q e T representam, respectivamente, a quantidade de $'F's$ e a quantidade de símbolos terminais presentes em $regra$. O valor de S_r retornado por essa equação, representando a quantidade de símbolos terminais que estariam presentes na produção de $regra$ até o nível n, deve ser comparado com o total de símbolos terminais contados na cadeia w. Caso os valores sejam iguais, a cadeia contida em $regra$ representará a solução do problema, caso contrário, o algoritmo passará para a próxima iteração, se houver, ou será encerrado retornando w. Dessa forma, esses procedimentos asseguram a consideração final feita em (iii).

As etapas para validação da regra encontrada, realizando a projeção da quantidade de símbolos terminais que ela pode produzir, estão organizadas e inclusas na função ***isPossible()*** presente na linha 9 do Algoritmo 3.

Algoritmo 3 - Problema Inverso

```
Entrada
   w: cadeia de caracteres
Saída
   regra: solução encontrada
Início
1   Ft←countSimbol('F', w)
2   se Ft>1 faça
3      para Fq←2 até int(sqrt(Ft)) faça
4           n←ln(Ft)/ln(Fq)
5           se isInteger(n) então
6              s← slice(w, Ft, Fq)
7              se replace(regra,s,w)=Fq então
8                 regra←terminalClear(regra, n)
9                 se isPossible(regra,Fq,n, St) então
10                   retornar regra
11                fim do se
12             fim do se
13          fim do se
14     fim do para
15  fim do se
16  regra←w
17  retornar regra
Fim
```

Os parâmetros requeridos pela função ***isPossible()*** compreendem a cadeia obtida do processo anterior, os valores de F_q e n atuais, além do total de símbolos terminais presentes na cadeia w.

Esse algoritmo agrupa e formaliza todos os conceitos apresentados até agora através do algoritmo 1 e algoritmo 2, além disso representa uma síntese da metodologia proposta para resolver o Problema Inverso de Lindenmayer dentro das restrições já apresentadas.

3.3 RESULTADOS OBTIDOS

Com base nas considerações anteriores, efetuou-se a análise de cadeias de diferentes níveis de produção obtidas a partir de gramáticas conhecidas. Na seqüência, são apresentadas, no Quadro 10, três cadeias distintas (w_1, w_2, w_3) para exemplificar, de forma resumida, a aplicação da metodologia proposta neste trabalho.

Quadro 10 - Cadeias utilizadas nos testes

Cadeia w_1	**Regra Original:** $F[+F]F[-F]F$
Cadeia produzida para análise:	
F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]	
Cadeia: w_2	**Regra Original:** $++F-F++F-F$
Cadeia produzida para análise:	
++++++F-F++F-F-++F-F++F-F++++F-F++F-F-++F-F++F-F-++++F-F++F-F-++F-F++F-F++++F-F++F-F-++F-F++F-F++++F-F++F-F-++F-F++F-F++++F-F++F-F-++F-F++F-F++++F-F++F-F-++F-F++F-F++++F-F++F-F-++F-F++F-F	
Cadeia: w_3	**Regra Original:** $FF-[-F+F+F]+[+F-F-F]$
Cadeia produzida para análise:	
FF-[-F+F+F]+[+F-F-F]FF-[-F+F+F]+[+F-F-F]-[-FF-[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+[+F-F-F]]+[+FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]]	

As cadeias w_1, w_2 e w_3 são analisadas de forma individual, respectivamente nos itens (a), (b) e (c).

a) Procurando a regra *L-system* para a cadeia w_1:

Etapa 1: pontos de crescimento identificados pelos sombreamentos claros e escuros. Estes pontos correspondem à cadeia retornada pela função ***slice()*** da linha 6 do Algoritmo 3;

```
F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-
F]F[+F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-
F]F]F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-
F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-
F]F]F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F
```

Etapa 2: pontos de crescimento substituídos por 'F'. Esta é a cadeia retornada pela função ***replace()***;

```
F[+F]F[-F]F
```

Etapa 3: A função ***terminalClear()*** verifica que não é necessário limpar nenhum símbolo terminal. Na seqüência, a função ***isPossible()*** retorna ***true*** pois verifica que a cadeia da etapa 2 pode produzir 186 símbolos terminais, precisamente a mesma quantidade encontrada em w_1. É observável também que a cadeia obtida é exatamente igual à regra utilizada na síntese de w_1, além corresponder à regra da gramática G_8.

Regra encontrada para w_1: `F[+F]F[-F]F`

b) Procurando a regra *L-system* para a cadeia w_2:

Etapa 1: pontos de crescimento identificados pelos sombreamentos;

```
++++++F-F++F-F-++F-F++F-F++++F-F++F-F-++F-F++F-F-++++F-F++F-F-++F-
F++F-F++++F-F++F-F-++F-F++F-F++++++F-F++F-F-++F-F++F-F++++F-F++F-F-
++F-F++F-F-++++F-F++F-F-++F-F++F-F++++F-F++F-F-++F-F++F-F
```

Etapa 2: pontos de crescimento substituídos por 'F's;

```
++++++F-++++F++++++F-++++F
```

Etapa 3: a função ***TerminalClear()*** verifica que devem ser removidos 4 caracteres terminais antes de cada 'F' presente na cadeia resultante da etapa anterior. Como existem 6 caracteres acumulados antes do primeiro 'F', e o nível de produção é indicado por $n=3$, basta efetuar a operação indicada pela equação 7: $(T_e/n)\cdot(n-1)=(6/3)\cdot(3-1)=4$, o que indica que devem ser removidos 4 caracteres antes de cada ocorrência de 'F'. A função também verifica que depois de cada 'F' não deve ser removido nenhum caractere pois não há nenhum símbolo depois do último 'F'. O sombreamento indica os símbolos que serão removidos.

```
++++F-++++F++++F-++++F
```

Etapa 4: A função ***isPossible()*** retorna ***true*** pois verifica que a cadeia resultante da etapa 3 pode produzir exatamente os mesmos 126 símbolos terminais contados em w_2. A cadeia obtida também é idêntica à regra utilizada para síntese dessa cadeia.

Regra encontrada para w_2: `++F-F++F-F`

c) Procurando a regra *L-system* para a cadeia w_3:

Etapa 1: pontos de crescimento identificados;

```
FF-[-F+F+F]+[+F-F-F]FF-[-F+F+F]+[+F-F-F]-[-FF-[-F+F+F]+[+F-F-F]+FF-
[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+[+F-F-F]]+[+FF-[-F+F+F]+[+F-F-F]-FF-[-
F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]]
```

Etapa 2: pontos de crescimento substituídos por 'F's;

```
FF-[-F+F+F]+[+F-F-F]
```

Etapa 3: A função ***isPossible()*** retorna ***true*** pois a cadeia obtida na etapa 2 pode produzir 108 símbolos terminais, exatamente a mesma contagem obtida na cadeia w_3.

Regra encontrada para w_3: `FF-[-F+F+F]+[+F-F-F]`

3.4 DISCUSSÃO DOS RESULTADOS

A metodologia proposta também foi aplicada sobre dois grupos de cadeias:

a) **Cadeias produzidas por 20 gramáticas, sendo algumas delas variações de gramáticas amplamente conhecidas:** nesse grupo, para cada gramática, foram geradas cadeias desde o nível um até o sexto nível de produção. Ao todo foram testadas 120 cadeias diferentes. No item (1) do Quadro 11, são apresentadas para exemplo, dez dessas gramáticas testadas, obtidas de PRUSINKIEWICZ e LINDENMAYER (1990);

b) **Cadeias produzidas por gramáticas geradas aleatoriamente:** foi desenvolvido um gerador de gramáticas aleatórias de tamanhos e complexidade variáveis. Foi gerado 1 milhão de gramáticas distintas. Cada uma dessas gramáticas foi utilizada para produzir uma cadeia ao quarto nível de iteração. Ao final, todas as cadeias produzidas, totalizando também 1 milhão, foram submetidas ao processo de reversão. No item (2) do Quadro 11, são apresentadas para exemplo, dez gramáticas geradas aleatoriamente;

Quadro 11 - Amostras de gramáticas testadas

1) Gramáticas obtidas de PRUSINKIEWICZ e LINDENMAYER (1990):	
1. F+F-F-F+F 2. F-F+F+FF-F-F+F 3. FF-F-F-F-F-F+F 4. FF-F+F-F-FF 5. FF-F-F-F-FF	6. F-F+F-F-F 7. F-FF--F-F 8. F[+F]F[-F]F 9. F[+F]F[-F][F] 10. FF-[-F+F+F]+[+F-F-F]
2) Gramáticas obtidas pelo gerador de gramáticas aleatórias:	
1. [+[[+F+F]+]] 2. +F+F+F+F+FFF++F++++F 3. FF+FFFFFFF+F+FFFF+FF+F+FF 4. [[[-F-[FF]-+F+]]] 5. [[-[[+[[++-[-[[F]-]]+++++FF+-+[++F+-]+[- F+][F]--[-F]+++-[+F--]-- +++]]-]]]]+	6. [[[F]]-]+[+[+F]][-+F-]+ 7. [+[F+FF]+F+++]++- 8. [+[F+]+F+[F]++[+[F-]-]---] 9. ---+F-[+F+-]-F+FF+++F-+F+- 10. [[[[[+F]]--+F-FF+F+F++FF-][[F-]-+[F]+]-]]

O algoritmo gerador de gramáticas aleatórias foi construído com base em uma gramática não determinística. Nessa gramática, definiram-se como regras, cadeias primitivas que pudessem originar uma regra *L-system* Livre de Contexto e Determinística. As cadeias primitivas utilizadas como regras foram: $[F]$, $+F$, $F+$, $-F$, $F-$ e FF. A partir de um axioma F, é realizado o processo de reescrita por algumas vezes (número também aleatório), não mais que 10 para evitar a formação de uma regra muito grande. Neste processo também é assegurado que nenhuma regra que contenha somente um $'F'$ seja retornada. O código desse gerador de gramática é apresentado a seguir, no Algoritmo 4. Esse código é executado de acordo com o número de gramáticas que se deseja gerar. Nesse algoritmo, nota-se a chamada da função ***aleatorioN()*** na linha 3, com o parâmetro 10, que retorna um número inteiro entre 1 e 10 gerado aleatoriamente. Esse número indica quantas vezes será efetuado o processo de reescrita na formação de uma regra. Já na linha 7, uma função similar é chamada para obter de forma aleatória uma cadeia primitiva contida em G, trata-se da função ***aleatorioG()***. O conjunto G contém as cadeias primitivas que serão utilizadas no processo de reescrita.

Algoritmo 4 - Gerador de Gramática Aleatória

```
Entrada
   G: conjunto de regras da gramática geradora
Saída
   regra: gramática gerada
Início
1   repetir
2     regra←'F';
3     para j←1 até aleatorioN(10) faça
4        str←Ø
5        para k←1 até tamanho(regra) faça
6           se regra[k]='F' faça
7             str←str+aleatorioG(G)
8           caso contrário
9             str←regra[k]
10          fim do se
11       fim do para
12    regra←str
13    fim do para
14  até que contagem('F',regra)>1
15  retornar regra
Fim
```

As regras obtidas pela execução do algoritmo 4 foram armazenadas em uma lista a fim de serem excluídas todas as regras duplicadas, mantendo apenas regras distintas.

As observações da aplicação da metodologia proposta sobre as cadeias do grupo (a) e (b) são:

- O algoritmo proposto para o processo de reversão conseguiu obter a regra *L-system* de todas as cadeias submetidas ao teste, tanto para cadeias do grupo (a) como para o grupo (b), retornando uma regra idêntica à utilizada na síntese da cadeia testada;

- O tempo de procura das regras *L-systems* dessas cadeias sofreu variações de acordo com o tamanho das cadeias submetidas. Contudo, a execução do algoritmo mostrou que a solução é encontrada dentro de um intervalo de tempo extremamente rápido. Por exemplo, na análise de uma das maiores cadeias testadas, contendo 7.872.138 caracteres (cerca de 1836 páginas em um editor de texto comum), a regra foi obtida em média dentro de 688 milissegundos;

- Outra informação relevante, que também pode ser obtida pelo algoritmo 3, é o nível de produção em que a regra encontrada pode reproduzir a cadeia analisada. Esse nível é dado pelo valor de n no instante em que uma regra é encontrada;

Todos os resultados apresentados foram obtidos em um sistema dotado com CPU de 1.800Mhz e 512MB de memória RAM com o sistema operacional Windows XP SP2, executando serviços comuns de uma estação de trabalho. Tanto o algoritmo de reversão, como o gerador de gramáticas aleatórias foi desenvolvido na linguagem Object Pascal no ambiente de desenvolvimento Delphi 7.

4 ESTUDOS SOBRE A OBTENÇÃO DE GRAMÁTICAS *L-SYSTEMS* LIVRES DE CONTEXTO E NÃO DETERMINÍSTICAS

Nesta seção são apresentadas as considerações sobre os estudos das Gramáticas *L-systems* Livres de Contexto e Não Determinísticas, objetivando a obtenção do formalismo gramatical a partir de amostras geradas por gramáticas conhecidas.

4.1 CARACTERIZAÇÃO DO PROBLEMA

As cadeias de caracteres produzidas pelas gramáticas *L-systems* Livres de Contexto e Não Determinísticas assumem uma conformação indeterminada, dado pelo comportamento aleatório das aplicações de suas regras de produção. Essa característica provoca dificuldades em qualquer processo que pretenda reconhecer os padrões de formação dessas cadeias. Normalmente, a pesquisa nos campos da Inferência Gramatical tem utilizado algoritmos genéticos, conforme foi possível observar nos trabalhos apresentados nas seções 2.3 e 2.4.

Diferentemente das cadeias produzidas pelas gramáticas determinísticas já estudadas, as cadeias produzidas por gramáticas não determinísticas podem levar a um processo de pesquisa muito custoso computacionalmente, já que uma mesma cadeia pode ter sido produzida por inúmeras e diferentes seqüências de derivações. Até mesmo estabelecer relações entre a quantidade de $'F's$ presentes e a ordem de crescimento da cadeia pode revelar um processo insustentável.

4.2 METODOLOGIA PROPOSTA

Em primeiro lugar, deve-se considerar que, diferentemente do problema considerando gramáticas determinísticas, aqui é necessário analisar

um conjunto de amostras resultantes de uma mesma gramática, e não apenas uma cadeia de caracteres. Dessa forma, optou-se por identificar uma metodologia que pudesse estabelecer relações estatísticas de ocorrências de determinadas subcadeias entre as amostras em análise. Entende-se que o elevado número de ocorrências de uma determinada subcadeia é um forte indicador de que essa subcadeia seja parte integrante das regras de produção que gerou as amostras.

Contudo, determinar essa subcadeia para realizar tal estatística é um dos grandes problemas no estudo para obtenção de gramáticas não determinísticas. A dificuldade está em identificar onde começa e termina uma subcadeia potencialmente candidata a uma das regras existentes na gramática utilizada na síntese das amostras, já que poderão existir várias regras em uma mesma gramática.

Assim sendo, alguns critérios devem ser adotados para que seja possível a procura dessas subcadeias:

i. quando houver seqüências de dois ou mais $'F's$ sucessivos nas cadeias analisadas, imediatamente é definida uma regra $F \rightarrow FF$. Esta definição serve para justificar a substituição de cada uma das ocorrências sucessivas por um único $'F'$, ocasionando a regressão de alguns pontos cadeia. Essas ocorrências apenas indicam a atuação do cursor gráfico em uma mesma direção, quantas vezes for o número de $'F's$ sucessivos, trazendo implicações apenas ao tamanho dos segmentos traçados. Este processo deve ser realizado imediatamente para tornar as cadeias mais simples e menos custosas de serem analisadas, conforme pode ser notado em um exemplo de regressão de uma cadeia qualquer, apresentada na Figura 15;
ii. deve ser considerada uma regra em potencial, toda subcadeia que for distinta, independente, e que contenha um ou mais $'F's$, além de outros símbolos, excluindo-se a subcadeia formada unicamente por $'F'$ sem qualquer outro símbolo. O significado de “distinta” implica que não há qualquer sentido na definição de duas ou mais regras

iguais. O termo “independente” implica no balanceamento adequado de regras formadas com os símbolos “[” e “]”. Esses símbolos não podem aparecer de forma ímpar ou na ocorrência inversa. Sempre que houver um símbolo (“[”) abrindo deverá haver outro (“]”) fechando, sempre aos pares, podendo até mesmo ocorrer aninhamentos.

iii. uma subcadeia não pode ser composta de partes já computadas nas subcadeias anteriores durante o processo de identificação das potenciais regras;

iv. por convenção, uma subcadeia que contenha um único 'F' apresentando os símbolos “[“ e “]”, entre outros, na sua constituição, deve ser escolhido com prioridade no momento da comparação com o próximo nível de formação das subcadeias. Se houver outras subcadeias com essa característica, será escolhida a que tiver uma contagem maior. Este critério foi adotado porque a subcadeia entre um abre e fecha colchetes muito provavelmente aparecerá um maior número de vezes do que quando for computada esta cadeia com os colchetes. Isto acontece por causa dos aninhamentos que ocorrem nas substituições sucessivas desta regra. Um exemplo disso pode ser visto na Figura 15, em que é possível notar que a cadeia '$+F$' aparece mais vezes que a cadeia '$[+F]$' devido ao aninhamento expresso na subcadeia '$[+F[+F]]$'.

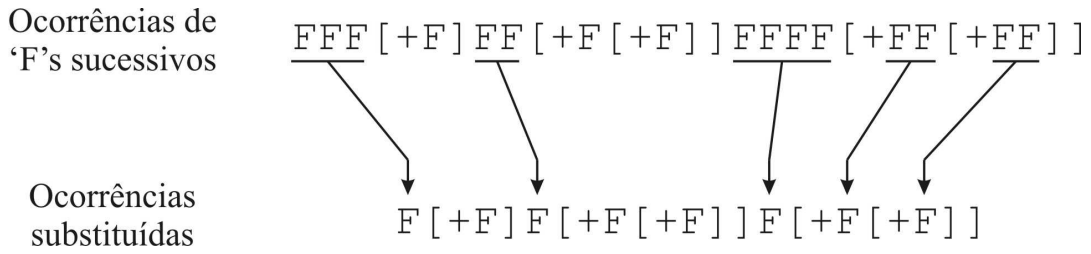


Figura 15 - Exemplo de regressão pela regra $F \rightarrow FF$

4.2.1 IDENTIFICAÇÃO DE REGRAS EM POTENCIAIS

No Quadro 12, é apresentada a gramática G_{10}, além de três cadeias distintas (z_1, z_2 e z_3) obtidas no quarto nível de produção.

Quadro 12 - Gramática *L-system* Livre de Contexto e Não Determinística

Gramática G_{10} $V = \{S, F\}$ $T = \{+, -, [,]\}$ $P = \{S \rightarrow F,$ $F \rightarrow FF,$ $F \rightarrow F[+F]$ $F \rightarrow F[-F]\}$	z_1	F[+F][-F[-F]][-FFF[-F]][+FF[-F[+F]][+F[-F][-F[+F]]]]
	z_2	F[+F][+FF]FFF[+F]F[+F][-F[+F]][+F[+F][+FF]]
	z_3	FFF[-F][-F[-F]F[+F]][+F[+F][+F[+F]][+F[+F][+F[+F]]]]

Antes de prosseguir para as etapas de identificação das regras, deve-se primeiramente efetuar a limpeza dos $'F's$ sucessivos, conforme esclarecido em (i). Na seqüência aparecem as cadeias z_1, z_2 e z_3 redefinidas, já com a limpeza efetuada.

z_1: F[+F][-F[-F]][-F[-F]][+F[-F[+F]][+F[-F][-F[+F]]]]

z_2: F[+F][+F]F[+F]F[+F][-F[+F]][+F[+F][+F]]

z_3: F[-F][-F[-F]F[+F]][+F[+F][+F[+F]][+F[+F][+F[+F]]]]

A seguir, são apresentadas as etapas necessárias para obtenção das subcadeias que podem corresponder às regras procuradas, responsáveis pela produção das cadeias z_1, z_2 e z_3.

Etapa 1: Percorrem-se as cadeias, da esquerda para a direita, adicionando-se em uma lista, todas as possíveis ocorrências distintas de subcadeias que contenham um único $'F'$ (note-se que os critérios ii e iii devem ser considerados nesta procura). Esse processo serve apenas para identificar

as possíveis partes que devem ser contabilizadas posteriormente. Somente de posse da lista completa é que o processo de contagem deve ser feito. O Quadro 13 apresenta essa lista e a contagem obtida para cada ocorrência. Também aparece destacado pelo sombreamento, o elemento da lista que será utilizado para comparação com o próximo nível de procura, conforme critério adotado em (v);

Quadro 13 - Lista de subcadeias com um $'F'$ (iteração 1)

Subcadeias	Contagem
[+F]	15
-F	11
[-F]	5
+F	22

Etapa 2: Executa-se o mesmo procedimento efetuado na etapa 1, contudo agora verificando as possíveis formações das subcadeias com dois $'F's$. O Quadro 14 apresenta a lista formada e a contagem obtida para cada ocorrência. O sombreamento indica o elemento que será comparado com o elemento escolhido na etapa anterior. Esta comparação determinará se a procura deverá prosseguir em busca de subcadeias com três ou mais $'F's$;

Quadro 14 - Lista de subcadeias com dois $'F's$ (iteração 1)

Subcadeias	Contagem
F[+F]	13
[-F[-F]]	2
[-F[+F]]	3
+F[-F]	1
[+F]F	2
+F[+F]	5
F[-F]	5
-F[-F]	3
[+F[+F]]	2

Após obtenção da lista do Quadro 14, deve-se comparar a proporção de grandeza entre a contagem assinalada nessa lista com a contagem assinalada na lista do Quadro 13. Divide-se o valor selecionado na contagem da lista atual (Quadro 14) pelo valor selecionado na contagem da lista anterior (Quadro 13). O valor obtido nessa divisão servirá de indicador de parada da primeira iteração. Esse indicador expressa quão próximo ou distante estão os valores analisados, revelando a necessidade de continuar ou não o processo de busca. Um valor muito alto indica que as subcadeias selecionadas ocorrem com freqüências muito parecidas, dificultando bastante a escolha entre uma ou outra, portanto, neste caso, a busca deve prosseguir. Já um valor mais baixo revela a distorção entre as duas subcadeias, indicando que a cadeia selecionada na lista anterior está mais apta a compor uma das regras procuradas, e a busca deve ser encerrada. Neste trabalho, foi adotado que a obtenção de valores inferiores a 0,5 ou 50% deve indicar o encerramento da procura na etapa corrente.

Sendo assim, 13 dividido por 15 resulta em 0,86 ou 86%. Isso implica em prosseguir mais uma etapa na procura de uma regra, agora contendo três $'F's$, e, neste momento, nenhuma regra será definida.

Etapa 3: Executa-se o mesmo procedimento efetuado na etapa 2, contudo agora verificando as possíveis formações das subcadeias com três $'F's$. O Quadro 15 apresenta a lista formada e a contagem obtida para cada ocorrência. O sombreamento indica o item que será comparado com o elemento escolhido na etapa anterior. Como há dois elementos com contagens iguais, pode-se escolher arbitrariamente qual será utilizado na comparação, já que suas contagens correspondem ao mesmo valor;

Assim, 2 dividido por 13 resultam em 0,15 ou 15%. Isto implica em encerrar o processo de busca e a subcadeia selecionada da etapa anterior é retornada como uma regra ($F \rightarrow F[+F]$).

Nesse ponto, como uma regra foi encontrada, devem-se regredir as cadeias usadas como amostra utilizando as regras já determinadas ($F \rightarrow FF$ e $F \rightarrow F[+F]$). O processo de regressão implica em substituir, nas

amostras, cada ocorrência de qualquer uma das regras por 'F', sucessivamente, até não haver mais substituições. Caso após o processo de regressão reste apenas um 'F' para cada amostra, não há mais regras a serem procuradas e a busca é encerrada. Caso ainda restem cadeias, o processo de busca deve ser reiniciado, desde a etapa 1, agora sobre as cadeias restantes, dando início à segunda iteração.

Quadro 15 - Lista de subcadeias com três '$F's$ (iteração 1)

Subcadeias	Contagem
[+F][-F[-F]]	1
+F[-F[+F]]	1
[-F][-F[+F]]	1
F[+F][+F]	2
F[+F]F	1
[+F][-F[+F]]	1
[+F[+F][+F]]	1
-F[-F]F	1
[+F][+F[+F]]	2

Na seqüência, são apresentadas as cadeias z_1, z_2 e z_3 redefinidas após a regressão efetuada pela aplicação das regras encontradas até agora, seguidas do reinício das etapas necessárias para obtenção da próxima regra.

z_1: F[-F[-F]][-F[-F]][+F[-F][+F[-F][-F]]]

z_2: F[-F][+F]

z_3: F[-F][-F[-F]F][+F]

Etapa 1: As subcadeias contendo um único 'F' (conforme critérios ii e iii) aparecem adicionadas no Quadro 16, juntamente com a contagem de cada

ocorrência. Também aparece destacado pelo sombreamento o elemento que será utilizado na comparação que indicará ou não o fim da busca.

Quadro 16 - Lista de subcadeias com um $'F'$ (iteração 2)

Subcadeias	Contagem
-F	11
[-F]	8
+F	4
[+F]	2

Etapa 2: As subcadeias contendo dois $'F's$ aparecem adicionadas no Quadro 17, juntamente com a contagem de cada ocorrência. Também aparece em destaque o elemento selecionado para comparação.

Quadro 17 - Lista de subcadeias com dois $'F's$ (iteração 2)

Subcadeias	Contagem
[-F[-F]]	2
+F[-F]	2
F[-F]	7
-F[-F]	3

Nesse ponto, efetuando-se a divisão de 7 (contagem atual) por 8 (contagem anterior) o resultado é 0,87 ou 87%. Isso significa que é necessário prosseguir mais uma etapa de procura, agora contendo três $'F's$. Nenhuma regra será definida.

Etapa 3: As subcadeias contendo três $'F's$ aparecem adicionadas no Quadro 18, juntamente com a contagem de cada ocorrência. Neste caso, como todas as contagens possuem o mesmo valor, pode-se escolher arbitrariamente qual será utilizado na comparação. Feito isso, efetua-se a divisão de 1 por 7, o que resulta em 0,14 ou 14%. Isto implica em encerrar o processo de busca, pois o indicador é menor que 50%. A subcadeia selecionada da etapa anterior é retornada como uma regra ($F \rightarrow F[-F]$).

Quadro 18 - Lista de subcadeias com três $'F's$ (iteração 2)

Subcadeias	Contagem
F[-F[-F]]	1
[+F[-F][-F]]	1
F[-F][+F]	1
[-F[-F]F]	1

Após a definição de mais uma regra, as amostras devem sofrer o processo de regressão, agora, aplicando-se a reversão de todas as regras encontradas. A Figura 16 apresenta a seqüência de reversão sofrida pela cadeia z_1 para exemplificar este processo.

Semelhantemente ao processo de regressão da cadeia z_1, as cadeias z_2 e z_3 também puderam ser revertidas até restarem apenas um $'F'$ em cada uma.

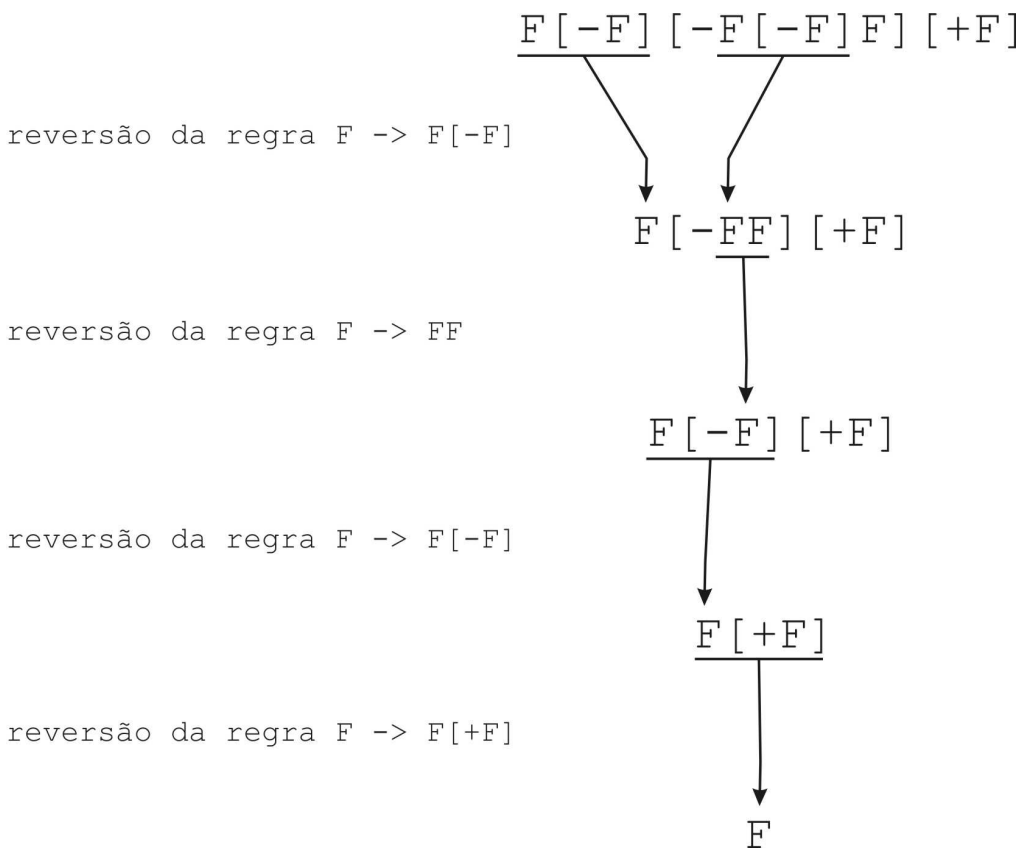


Figura 16 - Regressão da cadeia z_1

Logo, as regras encontradas ($F \rightarrow FF$, $F \rightarrow F[+F]$ e $F \rightarrow F[-F]$) não só são capazes de reverter as cadeias analisadas, como também são idênticas às regras utilizadas para sintetizar as amostras.

Deve-se destacar que aqui foram utilizadas cadeias de caracteres pequenas propositalmente para facilitar a explicação. Nos testes foram utilizadas cadeias maiores para que os resultados pudessem ser mais confiáveis.

4.3 RESULTADOS OBTIDOS

O Quadro 19 apresenta as gramáticas utilizadas nos testes.

Quadro 19 - Gramáticas utilizadas nos testes

a) Gramática G_{11} $V = \{S, F\}$ $T = \{+, -, [,]\}$ $P = \{S \rightarrow F,$ $F \rightarrow FF,$ $F \rightarrow F[+F]F[-F]\}$	**b) Gramática G_{12}** $V = \{S, F\}$ $T = \{+, [,]\}$ $P = \{S \rightarrow F,$ $F \rightarrow FF,$ $F \rightarrow F[+F]+F\}$
c) Gramática G_{13} $V = \{S, F\}$ $T = \{+, [,]\}$ $P = \{S \rightarrow F,$ $F \rightarrow FF,$ $F \rightarrow F[+F]$ $F \rightarrow F+F\}$	**d) Gramática G_{14}** $V = \{S, F\}$ $T = \{+, -, [,]\}$ $P = \{S \rightarrow F,$ $F \rightarrow FF,$ $F \rightarrow [+F]F$ $F \rightarrow [-F]F\}$
e) Gramática G_{15} $V = \{S, F\}$ $T = \{+, -, [,]\}$ $P = \{S \rightarrow F,$ $F \rightarrow FF,$ $F \rightarrow [+F]F$ $F \rightarrow F[-F]\}$	**f) Gramática G_{16}** $V = \{S, F\}$ $T = \{+, [,]\}$ $P = \{S \rightarrow F,$ $F \rightarrow FF,$ $F \rightarrow +[+F]F+\}$

Para cada gramática, foram geradas 5 amostras distintas obtidas entre o quarto e quinto nível de produção.

Nota-se ainda, que no processo de identificação das subcadeias que foram analisadas, optou-se por pesquisar somente a maior cadeia dada como amostra, no entanto, a contagem foi feita sobre as 5 amostras. Isto por que o tamanho da cadeia já é significativo e suficiente para execução do processo e identificação da lista de subcadeias.

Seguem-se adiante os resultados obtidos na aplicação da metodologia apresentada sobre as amostras.

a) Análise da Gramática G_{11} presente no item (a) do Quadro 19:

- **Amostras geradas no quarto nível de produção:**

1) FFF[+F]F[-F][+F[+F]F[-F][+FF]FF[-FF]]FFFF[-FF[+F[+F]F[-F]]FF[-FF]][+F[+F]F[-F][+FF]FF[-FF]FF[+F[+F]F[-F]]F[+F]F[-F][-F[+F]F[-F]]]F[+F]F[-F]FFF[+F]F[-F][+F[+F]F[-F]]F[+F]F[-F][-F[+F]F[-F]][-F[+F]F[-F][+FF]FF[-FF][+F[+F]F[-F]FF]F[+F]F[-F][+F[+F]F[-F]]FF[-F[+F]F[-F]][-F[+F]F[-F]F[+F]F[-F]]]
2) F[+F]F[-F][+F[+F]F[-F]]FF[-FF]F[+F]F[-F]F[+F]F[-F]F[+F]F[-F]F[+F]F[-F]FF[+FF]FF[-FF]
3) FF[+FF]FF[-F[+F]F[-F]][+F[+F]F[-F][+F[+F]F[-F]]F[+F]F[-F][-F[+F]F[-F]]]FFF[+F]F[-F][-F[+F]F[-F]FF]F[+F]F[-F]F[+F]F[-F]F[+F]F[-F][+FF]FF[-FF]
4) F[+F]F[-F]FFF[+F]F[-F][+F[+F]F[-F]]FF[-FF][+F[+F]F[-F][+F[+F]F[-F]]F[+F]F[-F][-F[+F]F[-F]]F[+F]F[-F]F[+F]F[-F]]FFF[+F]F[-F][+FF[+F[+F]F[-F]]F[+F]F[-F][-FF]]F[+F]F[-F][+FF]F[+F]F[-F][-F[+F]F[-F]][-FF[+FF]FF[-F[+F]F[-F]]][-F[+F]F[-F][+FF]FF[-F[+F]F[-F]]F[+F]F[-F]FF]
5) FFF[+F]F[-F]F[+F]F[-F]F[+F]F[-F]F[+F]F[-F]F[+F]F[-F][+F[+F]F[-F][+F[+F]F[-F]]F[+F]F[-F][-FF]]FF[+FF]FF[-FF][-FFF[+F]F[-F]]

- **Regressão por $F \rightarrow FF$:**

1) F[+F]F[-F][+F[+F]F[-F][+F]F[-F]]F[-F[+F[+F]F[-F]]F[-F]][+F[+F]F[-F][+F]F[-F]F[+F[+F]F[-F]]F[+F]F[-F][-F[+F]F[-F]]]F[+F]F[-F]F[+F]F[-F][+F[+F]F[-F]]F[+F]F[-F][-F[+F]F[-F]]][-F[+F]F[-F][+F[+F]F[-F]F]F[+F]F[-F][+F[+F]F[-F]]F[-F[+F]F[-F]]][-F[+F]F[-F]F[+F]F[-F]]]
2) F[+F]F[-F][+F[+F]F[-F]]F[-F]F[+F]F[-F]F[+F]F[-F]F[+F]F[-F]F[+F]F[-F]F[+F]F[-F]
3) F[+F]F[-F[+F]F[-F]][+F[+F]F[-F][+F[+F]F[-F]]F[+F]F[-F][-F[+F]F[-F]]]F[+F]F[-F][-F[+F]F[-F]F]F[+F]F[-F]F[+F]F[-F][+F]F[-F]
4) F[+F]F[-F]F[+F]F[-F][+F[+F]F[-F]]F[-F][+F[+F]F[-F][+F[+F]F[-F]]F[+F]F[-F][-F[+F]F[-F]]F[+F]F[-F]F[+F]F[-F]]F[+F]F[-

F][+F[+F[+F]F[-F]]F[+F]F[-F][-F]]F[+F]F[-F][+F]F[+F]F[-F][-F[+F]F[-F]][-F[+F]F[-F[+F]F[-F]]][-F[+F]F[-F][+F]F[-F[+F]F[-F]]F[+F]F[-F]F]

5) F[+F]F[-F]F[+F]F[-F]F[+F]F[-F]F[+F]F[-F]F[+F]F[-F][+F[+F]F[-F][+F[+F]F[-F]]F[+F]F[-F][-F]]F[+F]F[-F][-F+F]F[-F]]

- **Identificação e contagem das subcadeias:** No Quadro 20, aparecem em destaque os elementos que foram identificados e comparados para determinar o indicador de parada. A relação obtida entre a etapa 2 e a etapa 1 foi de 97%, já entre as etapas 3 e 2 foi de 93%. Entre as etapas 4 e 3 foi de 97%, porém entre as etapas 5 e 4, a relação foi de apenas 18%, o que significa que o item selecionado da etapa 4 corresponde a uma regra em potencial. Realizou-se o processo de reversão e cada amostra foi revertida em um único 'F'. Não foi necessária outra iteração. As regras da gramática procurada correspondem a $F \rightarrow FF$ e $F \rightarrow F[+F]F[-F]$.

Quadro 20 - Lista e Contagem das subcadeias encontradas nas amostras da gramática G_{11} (iteração 1)

Iteração 1									
Etapa 1		Etapa 2		Etapa 3		Etapa 4		Etapa 5	
[+F]	73	F[+F]	67	F[+F]F	67	F[+F]F[-F]	65	+F[+F]F[-F][+F]	2
[-F]	74	F[-F]	72	+F[+F]F	16	+F[+F]F[-F]	16	-F[+F[+F]F[-F]]	1
+F	90	+F[+F]	16	[-F][+F]F	6	[+F[+F]F[-F]]	10	F[+F[+F]F[-F]]	3
-F	90	[+F]F	73	-F[+F]F	15	[+F]F[-F]F	18	[-F][-F[+F]F[-F]]	6
		[-F]F	19	[-F]F[+F]	15	[-F[+F]F[-F]]	10	F[+F]F[-F]F	12
		-F[+F]	15			-F[+F]F[-F]	14	[-F][+F[+F]F[-F]]	7
								-F[+F]F[-F][+F]	2
								[+F[+F]F[-F]F]	1
								F[-F[+F]F[-F]]	4
								-F[+F]F[-F]F	2

b) Análise da Gramática G_{12} presente no item (b) do Quadro 19:

- **Amostras geradas no quarto nível de produção:**

1) F[+F]+F[+F[+F]+F]+FF[+FFFF]+FF[+FF]+F[+F]+F[+F[+F]+F[+F]+F]+F[+F]+F[+F[+F]+F[+FF]+F[+F]+F]+F[+F]+F[+FF]+FF]+FF[+FF]+F[+F]+F[+FFFF]+FF[+FF]+F[+F]+F

2) FFF[+F]+FF[+F]+FF[+F]+F[+FFF[+F]+F[+FFFF]+FF[+F[+F]+F]+FF]+F[+F]+F
F[+F]+F[+F[+F]+F[+F[+F]+F]+FF]+F[+F]+FFF

3) FFFFFF[+F[+F]+F]+FF[+F[+F]+F[+FF]+F[+F]+F[+F[+F]+F[+FF]+FF]+FF[+F[
+F]+F]+FF]+FF[+FF]+FFF[+F]+FFF

4) F[+F]+FFFFF[+F[+F]+F]+F[+F]+FFFFFFF[+F[+F]+F]+F[+F]+F

5) FF[+F[+F]+F]+FF[+FFF[+F]+F]+FF[+F[+F]+F]+F[+F]+FF[+F]+F[+FF]+F[+F]
+F[+F[+F]+FFF]+FFFF

- **Regressão por** $F \rightarrow FF$ **:**

1) F[+F]+F[+F[+F]+F]+F[+F]+F[+F]+F[+F]+F[+F[+F]+F[+F[+F]+F]+F[+F]+F[+
F[+F]+F[+F]+F[+F]+F]+F[+F]+F[+F]+F]+F[+F]+F[+F]+F[+F]+F[+F]+F[+F]+F[+F]+
F

2) F[+F]+F[+F]+F[+F]+F[+F[+F]+F[+F]+F[+F[+F]+F]+F]+F[+F]+F[+F]+F[+F]+F[+F[+
F]+F[+F[+F]+F]+F]+F[+F]+F

3) F[+F[+F]+F]+F[+F[+F]+F[+F]+F[+F]+F[+F[+F]+F[+F]+F]+F[+F[+F]+F]+F]+
F[+F]+F[+F]+F

4) F[+F]+F[+F[+F]+F]+F[+F]+F[+F[+F]+F]+F[+F]+F

5) F[+F[+F]+F]+F[+F[+F]+F]+F[+F[+F]+F]+F[+F]+F[+F]+F[+F]+F[+F]+F[+F[+
F]+F]+F

- **Identificação e contagem das subcadeias:** No Quadro 21, aparecem em destaque os elementos que foram identificados e comparados para determinar o indicador de parada. A relação obtida entre a etapa 2 e a etapa 1 foi de 100%, já entre as etapas 3 e 2 foi de 74%. Contudo, entre as etapas 4 e 3 a relação foi de 34%, o que indica que o item selecionado da etapa 3 corresponde a uma regra em potencial. Realizou-se o processo de reversão e cada amostra foi revertida em um único 'F'. Não foi necessária outra iteração. As regras da gramática procurada correspondem exatamente a $F \rightarrow FF$ e $F \rightarrow F[+F]+F$.

Quadro 21 - Lista e Contagem das subcadeias encontradas nas amostras da gramática G_{12} (iteração 1)

Iteração 1							
Etapa 1		Etapa 2		Etapa 3		Etapa 4	
+F	138	F[+F]+	51	F[+F]+F	38	+F[+F[+F]+F]+	9
[+F]	51	+F[+F]+	35	[+F[+F]+F]+	12	F[+F]+F[+F]+	13
		+F[+F]	16	[+F]+F[+F]+	13	+F[+F]+F[+F]+	12
				+F[+F]+F	35		
				[+F]+F[+F]	13		

c) Análise da Gramática G_{13} presente no item (c) do Quadro 19:

- **Amostras geradas no sexto nível de produção:**

1) F[+F]+F+FF[+F]+F+F+FF+FF[+F+F[+F+F]][+F+F[+F[+F]]+F[+F][+F[+F]]+F+F[+FF]F+F+F[+F]]F[+F][+F[+F]]+F[+F]FFF+F+FF[+FF+F+F]+F+FF+F+F+F+F[+F]+FF[+FF]F[+F]+F[+F]
2) F+FFF+F[+F]FFFF[+F+F][+F[+F]F[+F]][+F+F[+FF]+F+F+FFFF[+FF]+F+F[+FF]]FFF+FF[+F]+F[+F][+F+FF+F[+F[+F]+F[+F]]]F[+F]F+F[+F+FF+F]+FFF[+F]FF[+FF]
3) FF+F[+F][+FF+FF]F+F[+F[+F]]F+FF[+F]F[+F][+F+F]F+FF+F+F+F+F+F[+F+F+F[+F]]F[+F][+F[+F]][+F+FF+F]+FFFFF[+F]+F[+F]+F+F[+F+F]+F[+F]FFF+FF+F+F+F+F[+F]
4) F+FF[+F]F+F+F+F+FF+F[+F]+F[+F]+FF[+FF+F+FF[+F][+F+F]+F+F+FF+F[+F]+F+F]FF[+FF]+F+F[+F[+F]][+FF[+F[+F]]F[+F]F+F][+F[+F]+F[+F]+FF+F+FF+FFFF[+F]+FF]
5) F+F[+FF]FF[+F+F][+FF[+F[+F]][+F+F+F[+F]]]+F+FF[+F]+FFF+F+F+F+F+FF+F+F+FF[+F][+FF]+F[+F]FF+F+F+FF[+FF+FF]+F+FF[+F][+FF[+F[+F]]]+F[+F]+FF[+F+FFF]

- **Regressão por $F \rightarrow FF$:**

1) F[+F]+F+F[+F]+F+F+F+F[+F+F[+F+F]][+F+F[+F[+F]]+F[+F][+F[+F]]+F+F[+F]F+F+F[+F]]F[+F][+F[+F]]+F[+F]F+F+F[+F+F+F]+F+F+F+F+F+F[+F]+F[+F]F[+F]+F[+F]
2) F+F+F[+F]F[+F+F][+F[+F]F[+F]][+F+F[+F]+F+F+F[+F]+F+F[+F]]F+F[+F]+F[+F][+F+F+F[+F[+F]+F[+F]]]F[+F]F+F[+F+F+F]+F[+F]F[+F]
3) F+F[+F][+F+F]F+F[+F[+F]]F+F[+F]F[+F][+F+F]F+F+F+F+F+F+F[+F+F+F[+F]]F[+F][+F[+F]][+F+F+F]+F[+F]+F[+F]+F+F[+F+F]+F[+F]F+F+F+F+F+F[+F]
4) F+F[+F]F+F+F+F+F+F[+F]+F[+F]+F[+F+F+F[+F][+F+F]+F+F+F+F[+F]+F+F]F[+F]+F+F[+F[+F+F]][+F[+F[+F]]F[+F]F+F][+F[+F]+F[+F]+F+F+F+F[+F]+F]
5) F+F[+F]F[+F+F][+F[+F[+F]][+F+F+F[+F]]]+F+F[+F]+F+F+F+F+F+F+F+F+F[+F][+F]+F[+F]F+F+F+F[+F+F]+F+F[+F]+F[+F[+F]]]+F[+F]+F[+F+F]

Quadro 22 - Lista e Contagem das subcadeias encontradas nas amostras da gramática G_{13} (iteração 1)

Iteração 1					
Etapa 1		Etapa 2		Etapa 3	
[+F]+	20	F[+F]	59	F[+F]+F+	26
F+	91	+F+F	52	F[+F]+F	26
+F	216	[+F]+F+	9	+F[+F+F]	1
[+F]	60	F+F+	28	+F[+F[+F]]+	2
		[+F+F]	9	[+F][+F[+F]]+	23
		[+F[+F]]+	3	F+F[+F]	5
		[+F[+F]]	9	F[+F]F+F+	3
		[+F]F+	7	[+F+F+F]+	11
		F+F	63	F+F+F+	8
		+F+F+	21	[+F]+F[+F]	16
		[+F]+F	20	F+F+F	4
		[+F]F	13		

- **Regressão por $F \rightarrow FF$ e $F \rightarrow F+F$:**

1) F[+F]+F[+F]+F[+F[+F]][+F[+F[+F]]+F[+F][+F[+F]]+F[+F]F[+F]]F[+F][+F[+F]]+F[+F]F[+F]+F[+F]+F[+F]F[+F]+F[+F]
2) F[+F]F[+F][+F[+F]F[+F]][+F[+F]+F[+F]+F[+F]]F[+F]+F[+F][+F[+F[+F]+F[+F]]]F[+F]F[+F]+F[+F]F[+F]
3) F[+F][+F]F[+F[+F]]F[+F]F[+F][+F]F[+F[+F]]F[+F][+F[+F]][+F]+F[+F]+F[+F]+F[+F]+F[+F]F[+F]
4) F[+F]F[+F]+F[+F]+F[+F[+F][+F]+F[+F]+F]F[+F]+F[+F[+F]][+F[+F[+F]]F[+F]F][+F[+F]+F[+F]+F[+F]+F]
5) F[+F]F[+F][+F[+F[+F]][+F[+F]]]+F[+F]+F[+F][+F]+F[+F]F[+F]+F[+F][+F[+F]]]+F[+F]+F[+F]

- **Identificação e contagem das subcadeias:** No Quadro 23, aparecem em destaque os elementos que foram identificados e comparados para determinar o indicador de parada. A relação obtida entre a etapa 2 e a etapa 1 foi de 93%, contudo, entre as etapas 3 e 2 a relação foi de 25%, o que indica que o item selecionado da etapa 2 corresponde a uma regra em potencial. Realizou-se o processo de reversão e cada amostra foi revertida em um único 'F'. Não foi necessária outra iteração. As regras da gramática procurada correspondem exatamente a $F \rightarrow FF$ e $F \rightarrow F+F$ e $F \rightarrow F[+F]$.

Quadro 23 - Lista e Contagem das subcadeias encontradas nas amostras da gramática G_{13} (iteração 2)

Iteração 2					
Etapa 1		Etapa 2		Etapa 3	
[+F]	72	F[+F]+	23	F[+F]+F	17
+F	125	[+F[+F]]	12	+F[+F[+F]]	6
		[+F[+F]]+	3	+F[+F[+F]]+	1
		F[+F]	67	[+F][+F[+F]]+	2
		+F[+F]	43	F[+F]F	13
				[+F]+F[+F]+	8
				[+F]+F[+F]	16

d) Análise da Gramática G_{14} presente no item (d) do Quadro 19:

- **Amostras geradas no quinto nível de produção:**

1) [+[+[+[-[-F]F][+F]F][+FF][+F]F][-[-F]F]FF[-FF][+F]F][+[+[+F]F[-F]F]FF[+F]F][+[-[-F]F][+F]F][-[-F]F][-F]F

2) [-[-[-FF][+F]F][+[-F]F][-F]F][+[-[+F]F]FF][-FF][+F]F[-[+[-FF][+F]F][-FF]FF][-[+[+F]F]FF][+F]F[+F]F

3) [+[+[+[-F]F][+F]F]FF[+F]F[-[+F]F]FF[+FF][+F]F][+[+FF]FF[+F]FFF][+[-FF]FF][+[-F]F]FF

4) [+[-[+F]F][+F]F[+[-F]F][-F]F[-[+F]F][-F]F[+F]FFF][-[+F]F[+F]F][-[+F]F]FF[+[-FF]FF][+[-F]F][-F]F

5) [-[+[+FF[-F]F][-[-F]F][+F]F][+[+[-F]F][+F]F][-[+F]F]FF][+[+[+FF]FF][+FF][+F]F][-[+FF][+F]F][+FF][+F]F

- **Regressão por $F \rightarrow FF$:**

1) [+[+[+[-[-F]F][+F]F][+F][+F]F][-[-F]F]F[-F][+F]F][+[+[+F]F[-F]F]F[+F]F][+[-[-F]F][+F]F][-[-F]F][-F]F

2) [-[-[-F][+F]F][+[-F]F][-F]F][+[-[+F]F]F][-F][+F]F[-[+[-F][+F]F][-F]F][-[+[+F]F]F][+F]F[+F]F

3) [+[+[+[-F]F][+F]F]F[+F]F[-[+F]F]F[+F][+F]F][+[+F]F[+F]F][+[-F]F][+[-F]F]F

4) [+[-[+F]F][+F]F[+[-F]F][-F]F[-[+F]F][-F]F[+F]F][-[+F]F[+F]F][-[+F]F]F[+[-F]F][+[-F]F][-F]F

5) [-[+[+F[-F]F][-[-F]F][+F]F][+[+[-F]F][+F]F][-[+F]F]F][+[+[+F]F][+F][+F][+F]F

determinar o indicador de parada. A relação obtida entre a etapa 2 e a etapa 1 foi de 86%, porém entre as etapas 3 e 2, a relação foi de apenas 18%, o que significa que o item selecionado da etapa 2 corresponde a uma regra em potencial ($F \rightarrow [+F]F$). Realizou-se o processo de reversão considerando-se esta nova regra e as amostra não foram revertidas completamente, portanto, deve ser realizada uma nova iteração em busca de mais uma regra.

Quadro 24 - Lista e Contagem das subcadeias encontradas nas amostras da gramática G_{14} (iteração 1)

Iteração 1					
Etapa 1		Etapa 2		Etapa 3	
-[-F]	6	+[-[-F]F]	2	+[-[-F]F][+F]	2
[+F]	38	[+F]F	33	[+F][+F]F	5
[-F]	25	[+F][+F]	5	[-[-F]F]F	1
+[+F]	4	[-[-F]F]	5	[-F][+F]F	4
		F[-F]	3	+[+F]F[-F]	1
		+[+F]F	4	F[+F]F	6
		[-F]F	21	[-[-F]F][-F]	1
		F[+F]	7		

- **Regressão por $F \rightarrow FF$ e $F \rightarrow [+F]F$:**

1) [+[+[+[-[-F]F]F]F][-[-F]F]F[-F]F][+[+F[-F]F]F][+[-[-F]F]F][-[-F]F][-F]F
2) [-[-[-F]F][+[-F]F][-F]F][+[-F]F][-F]F[-[+[-F]F][-F]F][-F]F
3) [+[+[+[-F]F]F]F[-F]F][+F][+[-F]F][+[-F]F]F
4) [+[-F]F[+[-F]F][-F]F[-F][-F]F][-F][-F]F[+[-F]F][+[-F]F][-F]F
5) [-[+[+F[-F]F][-[-F]F]F][+[+[-F]F]F][-F]F][+F][-F]F

- **Identificação e contagem das subcadeias:** No Quadro 25, aparecem em destaque os elementos que foram identificados e comparados para determinar o indicador de parada. A relação obtida entre a etapa 2 e a etapa 1 foi de 94%, contudo, entre as etapas 3 e 2 a relação foi de 12%, o que indica que o item selecionado da etapa 2 corresponde a uma regra em potencial ($F \rightarrow [-F]F$). Realizou-se o processo de reversão e cada

amostra foi revertida em um único ‘F’. Não foi necessária outra iteração. As regras da gramática procurada correspondem exatamente a $F \rightarrow FF$ e $F \rightarrow [+F]F$ e $F \rightarrow [-F]F$.

Quadro 25 - Lista e Contagem das subcadeias encontradas nas amostras da gramática G_{14} (iteração 2)

Iteração 2					
Etapa 1		Etapa 2		Etapa 3	
-[-F]	6	+[-[-F]F]	2	+[+[-[-F]F]F]	1
[-F]	34	[-[-F]F]	6	[-[-F]F]F	4
+F	4	F[-F]	5	+[+F[-F]F]	2
		+F[-F]	2	[+[-[-F]F]F]	2
		-[-F]F	6	[-[-F]F][-F]	1
		[-F]F	32		

e) Análise da Gramática G_{15} presente no item (e) do Quadro 19:

- **Amostras geradas no quinto nível de produção:**

1) FF[-F[-F]][+F]F[-F[-F]][-[+F[-F][+F]F]FF[-FF]][-[+F[-F][-F[-F]][-F[-F][-[+F]F]]]F[-F]F[-F][+[+F]F][+F]F]

2) [+F[-F]FF][+FF][+F]FFFF[-F][-[+FF][+F]F][+[+[+[+F]F][+F]F]FF[-[+F]F]][+FF]FF[+F]F[-F[-F]]

3) [+FF]F[-F][-FF[+F]F][+FF][+F]F[+F]FFF[+FF[+F]F]FFFF[-[+FF[+F]F]FF[-F[-F]]]

4) [+[+F]F[-FF]]F[-F]F[-F][-FF[-FF][-[+F[-F]]F[-F]]][-[+F]F[-FF]F[-F][-FF][-[+[+F]F]F[-F][+FF]FF]]

5) [+FF]FF[-[+FF]F[-F]][+F]F[-FF][-[+FF][+F]F][+[+FF][+F]F]FF[-FF][-[+F]FFF[+[+F]F]FF]

- **Regressão por $F \rightarrow FF$:**

1) F[-F[-F]][+F]F[-F[-F]][-[+F[-F][+F]F]F[-F]][-[+F[-F][-F[-F]][-F[-F][-[+F]F]]]F[-F]F[-F][+[+F]F][+F]F]

2) [+F[-F]F][+F][+F]F[-F][-[+F][+F]F][+[+[+[+F]F][+F]F]F[-[+F]F]][+F]F[+F]F[-F[-F]]

3) [+F]F[-F][-F[+F]F][+F][+F]F[+F]F[+F[+F]F]F[-[+F[+F]F]F[-F[-F]]]

4) [+[+F]F[-F]]F[-F]F[-F][-F[-F][-[+F[-F]]F[-F]]][-[+F]F[-F]F[-F][-F][-[+[+F]F]F[-F][+F]F]]

5) [+F]F[-[+F]F[-F]][+F]F[-F][-[+F][+F]F][+[+F][+F]F]F[-F][-[+F]F[+[+F]F]F]

- Identificação e contagem das subcadeias: No Quadro 26, aparecem em destaque os elementos que foram identificados e comparados para determinar o indicador de parada. A relação obtida entre a etapa 2 e a etapa 1 foi de 85%, porém entre as etapas 3 e 2, a relação foi de apenas 13%, o que significa que o item selecionado da etapa 2 corresponde a uma regra em potencial ($F \rightarrow [+F]F$). Realizou-se o processo de reversão considerando-se esta nova regra e as amostra não foram revertidas completamente, portanto, deve ser realizada uma nova iteração em busca de mais uma regra.

Quadro 26 - Lista e Contagem das subcadeias encontradas nas amostras da gramática G_{14} (iteração 1)

Iteração 1					
Etapa 1		Etapa 2		Etapa 3	
-F	35	-F[-F]	7	F[-F[-F]]	4
[-F]	27	[+F]F	29	+F[-F][+F]	1
[+F]	34	[-F[-F]]	5	[-F][-F[-F]]	1
+F	40	+F[-F]	4	[-F][-[+F]F]	1
+[+F]	6	F[-F]	26	F[-F]F	4
		[-[+F]F]	2	[-F][+[+F]F]	1
		[+[+F]F]	4		

- **Regressão por $F \rightarrow FF$ e $F \rightarrow [+F]F$:**

1) F[-F[-F]]F[-F[-F]][-[+F[-F]F]F[-F]][-[+F[-F][-F[-F]][-F[-F][-F]]]F[-F]F[-F]F]
2) [+F[-F]F]F[-F][-F][+F[-F]]F[-F[-F]]
3) F[-F][-F]F[-F[-F[-F]]]
4) [+F[-F]]F[-F]F[-F][-F[-F][-[+F[-F]]F[-F]]][-F[-F]F[-F][-F][-F[-F]F]]
5) F[-F[-F]]F[-F][-F]F[-F][-F]

- **Identificação e contagem das subcadeias:** No Quadro 27, aparecem em destaque os elementos que foram identificados e comparados para determinar o indicador de parada. A relação obtida entre a etapa 2 e a etapa 1 foi de 81%, contudo, entre as etapas 3 e 2 a relação foi de 22%, o

que indica que o item selecionado da etapa 2 corresponde a uma regra em potencial ($F \rightarrow F[-F]$). Realizou-se o processo de reversão e cada amostra foi revertida em um único 'F'. Não foi necessária outra iteração. As regras da gramática procurada correspondem exatamente a $F \rightarrow FF$ e $F \rightarrow [+F]F$ e $F \rightarrow F[-F]$.

Quadro 27 - Lista e Contagem das subcadeias encontradas nas amostras da gramática G_{14} (iteração 2)

Iteração 2					
Etapa 1		Etapa 2		Etapa 3	
-F	44	[-F[-F]]	6	F[-F[-F]]	5
[-F]	33	+F[-F]	6	-[+F[-F]F]	1
+F	6	F[-F]	27	[-F][-F[-F]]	1
		-F[-F]	10	[-F[-F][-F]]	1
				F[-F]F	6

f) Análise da Gramática G_{16} presente no item (f) do Quadro 19:

- **Amostras geradas no quinto nível de produção:**

1) +[++[+FF]FF+FF+[+F]F+]+[++[++[+F]F+]+[+F]F++]+[++[+F]F+]+[+F]F++++
+[++[+F]F+FFFF+[+F]F+]+[+F]F+FF+[++[+F]F+]+[+F]F+++

2) +[++[++[++[++[+F]F+]+[+F]F++]+[+FF]FF++]+[+FFFF]+[+FF]FF+++]+[++[+
+[++[+F]F+]+[+F]F++]+[+FF]+[+F]F+++]+[++[++[+F]F+]FF+]+[+FF]+[+F]F
+++++

3) +[++[+FF+[+F]F++[+F]F++[+F]F+]FF+[+F]F++[+F]F++[+F]F++]+[++[++[+F]
F+]+[+F]F++]+[+FF]+[+F]F++++[+FFFF]+[++[+F]F+]FF+++

4) +[++[++[++[+F]F+]FF+]FF+[+F]F+++[++[+F]F+]+[+F]F+++[++[+F]F+]+[+F]
F++]+[++[++[++[+F]F+]FF+]FF+[+F]F++]FFFF+[+FF]FF+++

5) +[++[++[+F]F+FFFFFF]+[++[+F]F+]+[+F]F+++[+F]F+FF+]FF+[+F]F++[+FF]F
F++[+FFFF]+[+FF]FF+++

- **Regressão por $F \rightarrow FF$:**

1) +[++[+F]F+F+[+F]F+]+[++[++[+F]F+]+[+F]F++]+[++[+F]F+]+[+F]F+++++[+
+[+F]F+F+[+F]F+]+[+F]F+F+[++[+F]F+]+[+F]F+++

2) +[++[++[++[++[+F]F+]+[+F]F++]+[+F]F++]+[+F]+[+F]F+++]+[++[++[++[+F
]F+]+[+F]F++]+[+F]+[+F]F+++]+[++[++[+F]F+]F+]+[+F]+[+F]F+++++

3) +[++[+F+[+F]F++[+F]F++[+F]F+]F+[+F]F++[+F]F++[+F]F++]+[++[++[+F]F+
]+[+F]F++]+[+F]+[+F]F++++[+F]+[++[+F]F+]F+++

4) +[++[++[++[+F]F+]F+]F+[+F]F+++[++[+F]F+]+[+F]F+++[++[+F]F+]+[+F]F+
+]+[++[++[++[+F]F+]F+]F+[+F]F++]F+[+F]F+++

5) +[++[++[+F]F+F]+[++[+F]F+]+[+F]F+++[+F]F+F+]F+[+F]F++[+F]F++[+F]+[+F]F+++

- **Identificação e contagem das subcadeias:** No Quadro 28, aparecem em destaque os elementos que foram identificados e comparados para determinar o indicador de parada. A relação obtida entre a etapa 2 e a etapa 1 foi de 88%, porém entre as etapas 3 e 2, a relação foi de apenas 19%, o que significa que o item selecionado da etapa 2 corresponde a uma regra em potencial ($F \rightarrow +[+F]F+$). Realizou-se o processo de reversão considerando-se esta nova regra e as amostra foram revertidas completamente. Não foi necessária outra iteração. As regras da gramática procurada correspondem exatamente a $F \rightarrow FF$ e $F \rightarrow +[+F]F+$.

Quadro 28 - Lista e Contagem das subcadeias encontradas nas amostras da gramática *G*$_{14}$ (iteração 1)

Iteração 1					
Etapa 1		Etapa 2		Etapa 3	
++[+F]	24	++[+F]F+	20	++[+F]F+F+	3
F+	60	F+[+F]	8	++[++[+F]F+]+[+F]	6
[+F]	52	++[++[+F]F+]+	6	+[++[+F]F+]+[+F]	9
+[+F]	52	[+F]F++	25	+[+F]F+F+	4
F++	26	+[++[+F]F+]+	9	[++[+F]F+]+[+F]	9
F+++++	2	[+F]F+++++	2	++[+F]F+F+	3
F+++	12	+[+F]F+	46		
		[+F]F+++	11		

4.4 DISCUSSÃO DOS RESULTADOS

As impressões da aplicação da metodologia estudada até agora sobre as amostras produzidas por essas gramáticas são:

- As gramáticas testadas foram escolhidas por apresentarem alguma variação na formação das regras, contudo, optou-se por gramáticas de menor complexidade e com menor número de regras devido a maior parte do processo ser ainda manual;

- Foi possível obter regras das gramáticas testadas exatamente igual às gramáticas utilizadas nos testes;

- As amostras utilizadas foram produzidas entre o quarto e quinto nível de produção das gramáticas (em um exemplo foi utilizado o nível seis). Contudo, percebe-se a relação entre o número de amostras e o nível de produção. Quanto menor for o nível de produção, maior deverá ser o número de amostras, para garantir uma diversidade nas cadeias de modo que o processo possa ter maiores probabilidades de identificar uma subcadeia que corresponda a uma das regras procuradas. Já uma cadeia gerada em um alto nível de produção, pode apresentar um bom índice de probabilidade de ter suas regras encontradas.

Com este estudo, é possível dizer que a metodologia proposta é um caminho para a descoberta de uma metodologia definitiva para o Problema Inverso de Lindenmayer para gramáticas não determinísticas. Estudos mais exaustivos com a utilização de um número maior de exemplos de gramáticas ainda são necessários.

5 CONCLUSÕES

O objetivo norteador deste trabalho foi a obtenção de gramáticas *L-systems* a partir de cadeias de caracteres geradas em algum nível de produção de uma gramática *L-system* livre de contexto desconhecida. Foram consideradas, no processo de busca, as características de auto-similaridade e reescrita paralela desses sistemas. A motivação inculcada na execução deste trabalho foi prover subsídios para a obtenção de regras *L-systems* para a reconstrução de neurônios artificiais em ambiente de realidade virtual, além de contribuir diretamente para a solução do Problema Inverso de Lindenmayer.

A obtenção ou procura por uma gramática *L-system* é conhecida como o problema inverso de Lindenmayer e é uma particularidade dos problemas de inferência gramatical. Este trabalho apresentou uma proposta para solucionar o Problema Inverso de Lindenmayer em uma subclasse de gramáticas *L-systems*.

Foram estudadas propostas baseadas em metodologias e abordagens diferentes para a resolução do problema inverso de Lindenmayer. Embora em cada uma dessas propostas houvesse contribuições interessantes para resolução desse problema, em nenhuma delas foi encontrada uma metodologia que resultasse em uma solução descrita nos moldes de uma gramática formal, mesmo nos casos em que foram consideradas apenas gramáticas determinísticas semelhantes às gramáticas testadas neste trabalho.

Baseado nas características de cadeias de caracteres produzidas por gramáticas *L-systems* Livres de Contexto e Determinísticas conhecidas, foi proposto um algoritmo capaz de regressar essa cadeia de caracteres até sua regra geradora, idêntica à regra original usada na síntese da amostra. Foram testadas mais de 1.000.000 cadeias diferentes. Os testes realizados produziram resultados 100% exatos, em todos os testes foi possível obter uma regra que permitiu a reprodução da cadeia de caracteres analisada. Nota-se que a execução do algoritmo na obtenção dessas regras tem evidenciado um tempo de execução extremamente rápido. Em um dos maiores

testes realizados, na análise de uma cadeia com cerca de 7,8 milhões de caracteres, a solução foi encontrada em menos de 1 segundo.

Em relação às gramáticas *L-systems* Livres de Contexto e Não Determinísticas foram estudados métodos que permitiram obter as regras exatas das gramáticas testadas. Com esses estudos foi possível identificar alguns procedimentos capazes de selecionar subcadeias das amostras, potencialmente candidatas às regras procuradas. Os testes realizados apontam para resultados promissores.

Portanto, a metodologia proposta neste trabalho apresentou a possibilidade de obtenção de regras *L-systems* a partir de cadeias de caracteres descritoras dos objetos em estudo.

Trabalhos futuros envolverão a continuidade dos estudos sobre a obtenção de regras para gramáticas *L-systems* Livres de Contexto e Não Determinísticas, visando a automação do processo e o desenvolvimento de método para verificação dos resultados. Também há possibilidades de estudos sobre métodos de obtenção automática das cadeias, a partir do objeto conhecido, ou ainda, a obtenção de métodos que suportem outros grupos de gramáticas *L-systems*, além das classes suportadas por esta proposta, como por exemplo, gramáticas estocásticas.

REFERÊNCIAS BIBLIOGRÁFICAS

ABELSON, H.; diSESSA A. A. **Turtle geometry**. M.I.T. Press, Cambridge, 1982.

AHO, I.; KEMPPAINEN, H.; KOSKIMIES, K.; MÄKINEN, E.; NIEMI, T. **Searching neural network structures with L Systems and genetic algorithms**. Tampere: Department of Computer Science - University of Tampere, 1997. Relatório técnico A-1997-15.

AONO, M.; KUNII, T. L. Botanical tree image generation. **IEEE Computer Graphics and Applications**, Tokyo, v. 4, n. 5, p. 10-34, May, 1984.

BERTERO, M.; BOCCACCI, P. **Introduction to inverse problem in imaging**. Institute of Physics Publishing, 1998.

BOER, M. J. M.; FRACCHIA, F. D.; PRUSINKIEWICZ, P. Analysis and simulation of the development of cellular layers. In: LANGTON, C. G. et al. **Artificial Life II**, SDI Studies in the Sciences of Complexity, Addison-Wesley, v. X, p. 465-483, 1991.

BRONS, R. Theoretical and linguistic method for describing straight lines. In: EARNSHAW, R. A. **Algorithms for Computer Graphics**, Spinger-Verlag, Berlin Heidelberg, p. 19-57, 1995.

CHALMOND, B. **Modeling and inverse problems in image analysis**. Springer, 2003.

COELHO, R. C.; JAQUES, O. Generating three-dimensional neural cells based on bayes rules and interpolation with thin plate splines. **Lecture Notes in Computer Science - Progress in Pattern Recognition, Speech and Image Analysis**, v. 2905, 2003, p. 675-682.

COHEN, D. I. A. **Introduction to computer theory**. 2. ed. New York: John Wiley & Sons, 1997.

COSTA, E. L.; LANDRY, J. A. Generating grammatical plant models with genetic algorithms. **Proceeding of the International Conference on Adaptive and Natural Computing Algorithms**. Coimbra, Portugal: March 21-23. SpringerWien, New York. p. 230-234, 2005.

COWELL, J. R. Syntactic pattern recognizer for vehicle identification numbers. **Image and Vision Computing**, v. 13, n. 1, p. 13-19, 1995.

FRIJTERS, D.; LINDENMAYER, A. A model for the growth and flowering of aster novae-angiae on the basis of table (1,0) *L-systems*. In: ROZENBERG, G.; SALOMAA, A. **L Systems**, Lecture Notes in Computer Science, Springer-Verlag, v. 15, p. 24-52, 1974.

GRATE, L.; HERBSTER, M.; HUGHEY, R.; HAUSSLER, D.; MIAN, I. S.; NOLLER, H. RNA modeling using gibbs sampling and stochastic context free grammars. In: **Proc. of Second Int. Conf. on Intelligent Systems for Molecular Biology**, Menlo Park, CA: AAAI/MIT Press, August 1994.

HAMILTON, P. A language to describe the growth of neuritis. **Biological Cybernetic**, v. 68, p. 559-565, 1993.

HIGUERA, C. de La. Current trends in grammatical inference. In: F.J.F. et al. **Advances in Pattern Recognition**, Joint IAPR International Workshops SSPR+SPR2000, Lecture Notes in Computer Science, Springer-Verlag, v. 1876, p. 28-31, 2000.

HIGUERA, C. de La. A bibliographical study of grammatical inference**. Pattern Recognition**. n. 38, p. 1332-1348, 2005.

HOGEWEG, P.; HESPER, B. A model study no biomorphological description. **Pattern Recognition**, v. 6, p. 165-179, 1974.

HOLLIDAY, D. J.; SAMAL, A. Object recognition using *L-system* fractals. **Pattern Recognition Letters**, v. 16, p. 33-42, Jan. 1995.

HORNBY, G. S.; POLLACK, J. B. Evolving *L-systems* to generate virtual creatures. **Computers and Graphics**, v. 6, n. 25, p. 1041-1048, 2001.

KALAY, A.; PARNAS, H.; SHAMIR, E. Neuronal growth via hybrid system of self-growing and diffusion based grammar rules. **I Bulletin of Mathematical Biology**, v. 57, n. 2, p. 205-227, 1995.

KASHYAP, R.L. Syntactic decision rules for recognition of spoken words and phrases using stochastic automaton. **IEEE Transactions on Pattern Analysis and Machine Intelligence,** v. 1, n. 2, p. 154–163,1979.

KOZA, J. R. Discovery of rewrite rules in Lindenmayer systems and state transition rules in cellular automata via genetic programming. **Symposium on Pattern Formation (SPF-93)**, Claremont, California, p. 1-19, 1993.

LINDENMAYER, A. Mathematical models for cellular interaction in development. **Journal of Theoretical Biology**, v. 18, p. 300-315, 1968.

MANDELBROT, B. B. **The fractal geometry of nature**. W. H. Freeman, San Francisco, 1982.

MCCORMICK, B. H.; MULCHANDANI, K. *L-system* modeling of neurons. **Visualization in Biomedical Computing Conference Proceedings**, 1994.

MENEZES, P. F. B. **Linguagens formais e autômatos**. 4 ed. Porto Alegre : Instituto de Informática da UFRGS, Editora Sagra Luzzatto, 2001.

MENKE, W. **Geophysical data analysis**: discrete inverse theory. Academic Press, 1989.

NATTERER, F.; WÛBBELING, F. **Mathematical methods in image reconstruction**. Society for Industrial and Applied Mathematics, 2001.

PAPERT, S. **Logo: computadores e educação**. São Paulo: Brasiliense, 1986

PARKER, R. L. **Geophysical inverse theory**. Princeton University Press, 1994.

PRUSINKIEWICZ, P. Graphical applications of *L-systems*. In: **Proceedings of Graphics Interface '86 - Vision Interface '86**, p. 247-253, 1986a.

PRUSINKIEWICZ, P. Score generation with *L-systems*. **Proceedings of the 1986 International Computer Music Conference**, p. 455-457, 1986b.

PRUSINKIEWICZ, P.; LINDENMAYER, A. **The algorithmic beauty of plants.** Springer-Verlag, New York, 1990.

RAMM, A. G. **Inverse Problems**: mathematical and analytical techniques with applications to engineering. Springer, 2005.

RUDOLPH, S.; ALBER, R. An evolutionary approach to the inverse problem in rule-based design representations. **Proceedings 7th International Conference on Artificial Intelligence in Design**, Cambridge University, Cambridge, UK, July 15-17th, 2002.

STEVENS, R. J.; LEHAR, A. F.; PERSTON, F. H. Manipulation and representation of multidimensional image data using the Peano scan. **IEEE Transactions on Pattern Analysis and Machine Intelligence,** v. 5, n. 5, p. 520–526,1983.

SAIDI, A. S.; TAYEB-BEY, S. Grammatical inference in document recognition: In: **Grammatical Inference**, Lecture Notes in Computer Science, London, Springer-Verlag, v. 1433, p. 175-186, 1998.

SAGAN, H. **Space-Filling Curves**, Springer-Verlag, 1994.

SAKAKIBARA, Y.; BROWN, M.; HUGHEY, R.; MIAN, I. S.; SJOLANDER, K.; UNDERWOOD, R.; HAUSSLER, D. Stochastic context-free grammars for tRNA modeling. **Nuclear Acids Research**. v. 22, p. 5112–5120, 1994.

SAKAKIBARA, Y. Grammatical inference: an old and new paradigm. **Lecture Notes Artificial Intelligence**, v. 997, p. 1-24, 1995.

SAMAL, A.; PETERSON, B.; HOLLIDAY, D. J. Recognition of plants using a stochastic *L-system* model. In: **Journal of Electronic Imaging**, SPIE, v. 11, part 1, p. 50-58, 2002.

SCHWEHM, M.; OST, A. Inference of stochastic regular grammars by massively parallel genetic algorithms. In: ESHELMAN, L. J. **Genetic Algorithms: Proceedings of the sixth Int. Conf. (ICGA95)**, Morgan Kaufmann Publishers, 1995, p. 520-527.

SEARLS, D. The computational linguistics of biological sequences. In: HUNTER, L. **Artificial Intelligence and Molecular Biology**, AAAI Press, 1993, p. 47-120.

SHLYAKHTER, I.; ROZENOER, M.; DORSEY, J.; TELLER, S. Reconstruction of 3D tree model from instrumented photographs. **IEEE Computer Graphics and Applications**, p. 53-61, 2001.

SIROMONEY, R.; SUBRAMANIAN, K. G. Space-filling curves and infinite graphs. In: H. EHRIG, H.; NAGL, M.; ROZENBERG, G. Graph grammars and their application to computer science, **Second International Workshop,** Lecture Notes in Computer Science, Springer-Verlag, Berlin, v. 153, p. 380-391, 1983.

SMITH, A. R. Plants, fractals, and formal languages. **Computer Graphics,** v.18, n. 3, p. 1-10, 1984.

SUDKAMP, T. P. **Languages and machines: an introduction to the theory of computer science**. 2. ed. Addison-Wesley, 1997.

SZILARD, A. L.; QUINTON, R. E. An interpretation for DOL Systems by computer graphics, **The Science Terrapin**, n. 4, p. 8-13, 1979.

TARANTOLA, A. **Inverse problem theory and methods for model parameter**. Society for Industrial and Applied Mathematics, 2005.

TARANTOLA, A. Popper, Bayes and the inverse problem. **Nature Physics**, v. 2, n. 8, p. 492-494, 2006.

UHLMANN, G. **Inside out: inverse problems and applications**. Mathematical Sciences Research Institute Publications, 2003.