Obtenção de regras L-system a partir de cadeias de caracteres

Autores

Edmar Santos

Orientador

Regina Celia Coelho

1. Introdução

L-system, Lindenmayer system ou sistemas paramétricos [1], trata-se de uma gramática formal concebida por Aristid Lindenmayer em 1968, a partir da gramática de Noam Chomsky. Tal gramática foi originalmente introduzida e usada para descrever o processo de crescimento de plantas [1,2], porém também permite a geração de fractais [2], modelagem de tráfego de redes e outros. Lindenmayer observou que tal modelo realmente poderia representar a reprodução celular de um organismo, resultando no seu crescimento, e que esta proposta poderia ser facilmente implementada em um computador. Desta forma, assumindo-se um axioma e uma determinada regra sobre um dado alfabeto, tem-se a possibilidade de reconstruir um modelo pela interpretação gráfica da cadeia de caracteres resultante. Esta interpretação baseia-se no paradigma da tartaruga (da linguagem LOGO), em que cada símbolo tem um efeito na reconstrução gráfica do modelo. No traçado do modelo a ser reconstruído, o caractere ‘F’ representa traçado de uma reta, os símbolos ‘+’ e ‘–’ representam girar para a direita ou esquerda, respectivamente, sobre o eixo z e os símbolos ‘[’ e ‘]’ acumulam a atual posição e retomam a posição acumulada, respectivamente. Na Figura 1 vê-se um exemplo de modelo gerado pela gramática 1 (considere que S é o axioma).

Gramática 1:

S->F

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

A Figura 1 foi reconstruída pela interpretação gráfica da gramática 1 após o 3º. nível de iteração.

A dificuldade está na representação destes modelos por meio de uma gramática L-system. Tendo-se a cadeia gerada por um determinado nível de produção é possível obter-se a regra geradora de tal cadeia? A proposta deste projeto é a obtenção da gramática L-system que represente devidamente estes modelos.

2. Objetivos

O projeto descrito neste trabalho tem como objetivo o desenvolvimento de uma metodologia que consiga obter a regra geradora (ou regras geradoras) de uma imagem (normalmente imagens que possam ser descritas por uma gramática). Esta regra deverá estar na forma de uma gramática L-system. Pretende-se também que a cadeia descritora desta imagem, seja obtida automaticamente por métodos de rastreamento de imagem. Uma grande vantagem desta proposta, além permitir obter a gramática geradora do modelo, é poder gerar diferentes formas com a mesma gramática, porém estatisticamente semelhantes ao modelo, desde que a L-system definida seja estocástica.

3. Desenvolvimento

Os procedimentos adotados até o momento consistem na análise e comparação de gramáticas L-system conhecidas, cujo axioma seja definido por apenas um caractere ‘F’, a fim de se encontrar características ou padrões nas cadeias de caracteres geradas em vários níveis de iteração que possam levar à uma definição da gramática. Por exemplo, considerando os três primeiros níveis de iteração dados pela gramática 1, ter-se-á as seguintes cadeias de caracteres:

Nível 1: F[+F]F[-F]F

Nível 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

Nível 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

Para cada nível de iteração, realizou-se a contagem de caracteres ‘F’ presentes em toda a cadeia:

Nível 1: 5 Fs

Nível 2: 25 Fs

Nível 3: 125 Fs

Portanto, a ordem de crescimento dos níveis de iteração segue na forma exponencial dada pela quantidade de caracteres ‘F’ presentes na regra de produção em ‘F’ (F_r) elevado ao nível de iteração (n), como na fórmula a seguir:

$F_s=F_r^n$

sendo:

F_s= quantidade de caracteres ‘F’ da cadeia gerada

F_r= quantidade de caracteres ‘F’ da regra de produção em ‘F’

n = nível de iteração

e F_s, F_r e n pertencem à N^*;

Assim, para a gramática 1, o total de caracteres ‘F’ encontrado em cada um dos níveis de iteração poderá ser obtido como:

Nível 1: $F_s = F_r^n = 5^1 = 5$ caracteres ‘F’ na cadeia para n = 1

Nível 2: $F_s = F_r^n = 5^2 = 25$ caracteres ‘F’ na cadeia para n = 2

Nível 3: $F_s = F_r^n = 5^3 = 125$ caracteres ‘F’ na cadeia para n = 3

Este mesmo padrão foi encontrado em outras gramáticas com outros níveis de produção.

Outra observação feita está no formato encontrado em todas as cadeias de todos os níveis de iteração. Estas cadeias seguem o formato ditado pela própria regra de produção em ‘F’. Na regra de produção em ‘F’ da gramática 1 existem cinco caracteres ‘F’ separados entre si por alguns símbolos. Observe esta mesma regra com os espaços em torno dos caracteres ‘F’ expandidos, para que se possa melhor visualizar as partes que compõem a regra:

F [+ F] F [- F] F

Veja que os grupos de símbolos ‘[+’, ‘]’, ‘[-’, ‘]’ que estão entre os caracteres ‘F’ da regra de produção são justamente os caracteres terminais da gramática 1. Estes grupos serão chamados aqui de "separador(es)". Com esta expansão dos espaços, pode-se perceber que esta regra é composta de 9 partes, sendo 5 partes

compostas apenas de ‘F’ e 4 partes compostas pelos separadores. Distribuindo-se cada parte, uma em cada linha, têm-se:

1 F

2 [+ (separadores)

3 F

4] (separador)

5 F

6 [- (separadores)

7 F

8] (separadores)

9 F

Pela própria característica das gramáticas L-system, cada caractere ‘F’ desta regra de produção será substituída pela cadeia de caracteres definidas na própria regra, sucessivamente, por n iterações. Isto significa que no lugar de cada caractere ‘F’ será injetado uma nova cadeia de caracteres, contudo, esta injeção de caracteres não modifica os separadores nem suas posições, independente do nível de iteração. No exemplo anterior, esta injeção de caracteres ocorrerá somente nas partes 1, 3, 5, 5 e 9, onde justamente estão posicionados os ‘Fs’, caracteres não-terminais da gramática 1, e estas partes sempre serão iguais. Assim, é possível concluir que para qualquer nível de iteração desta gramática, sempre haverão 9 partes, sendo 5 partes iguais, seja qual for o tamanho da nova cadeia, e 4 partes de separadores.

1 (cadeia de caracteres resultantes pela injeção em ‘F’ de após n iterações)

2 [+ (separadores)

3 (cadeia de caracteres resultantes pela injeção em ‘F’ de após n iterações)

4] (separador)

5 (cadeia de caracteres resultantes pela injeção em ‘F’ de após n iterações)

6 [- (separadores)

7 (cadeia de caracteres resultantes pela injeção em ‘F’ de após n iterações)

8] (separadores)

9 (cadeia de caracteres resultantes pela injeção em ‘F’ de após n iterações)

A cadeia de caracteres resultante após o segundo nível de iteração da gramática 1 pode ser representado como:

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

2 [+ (separadores)

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

4] (separador)

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

6 [- (separadores)

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

8] (separadores)

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

Já a cadeia de caracteres resultante após o terceiro nível de iteração da gramática 1 pode ser representado desta forma:

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 [+ (separadores)

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] (separador)

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

6 [- (separadores)

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

8] (separadores)

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

Como a regra de produção da gramática 1 apresenta 5 ‘Fs’ geradores, então o formato de todas as cadeias em qualquer nível de iteração também terá 5 partes iguais. Portanto, reconhecer o formato de uma regra a partir de uma cadeia significa saber quantas partes são resultantes dos ‘Fs’ geradores e quais os seus separadores.

Entendendo-se que a ordem de crescimento provocada em qualquer nível de iteração é de ordem exponencial, procura-se um número (F_r) que efetuado seu produto por ele mesmo n vezes resulte na quantidade de ‘Fs’ encontrada numa cadeia de caracteres (F_s) de um determinado nível de iteração. Neste processo, como apenas F_s é conhecido, então propõe-se que seja feita uma busca iterativa para se encontrar F_r e n.

Como F_r=1 sempre resultará F_s=1, então a primeira busca de F_r e n ocorrerá para F_r=2, em que a regra geradora teria supostamente 2 ‘Fs’ presentes; a próxima busca ocorreria para F_r=3, e assim por diante. Assim, F_s e supostamente F_r serão conhecidos, faltando então encontrar n de forma que $F_s=F_r^n$. Isso trata-se de uma função logarítmica representada na fórmula:

$$(Ln\ F_s / Ln\ F_r) = n$$

Se o resultado obtido por esta expressão for um número inteiro, então a iteração tem o seu fim. Caso contrário, uma nova iteração deverá ocorrer, incrementando-se a base logarítmica F_r (neste caso, F_r=3), e assim por diante. A definição do limite máximo para estas iterações pode ser determinada pela parte inteira da própria raiz quadrada de F_s, pois jamais existirá uma exponenciação em F_r , maior que 1 que produza F_s, cujo expoente seja menor que 2. Caso n não seja encontrado dentro destes limites, F_s = F_r pois n será igual a 1. Para sistematizar estas conclusões, o seguinte pseudocódigo pode ser apresentado:

```
para i=2 até int(sqrt(Fs)) faça

n=ln(Fs)/ln(i)

se n for inteiro então

retornar i, n

retornar Fs,1
```

Aplicando este pseudocódigo para a cadeia de caracteres do segundo nível de iteração da gramática 1, em que F_s=25, a iteração em i será encerrada após a 4ª. execução (encontrou um n inteiro) e o código retornará o valor de F_r e n, respectivamente 5 e 2. Isto representa que esta cadeia foi gerada por n=2 iterações e a regra de produção apresenta 5 ‘Fs’ na sua formação, o que implica que a cadeia resultante deverá apresentar 5 partes iguais intercaladas pelos separadores. Um algoritmo de casamento de cadeias adaptado poderia encontrar estas partes iguais que se repetem 5 vezes na cadeia resultante, e quebrá-las em função destas 5 partes, tendo como resultado a seguinte distribuição:

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

2 [+
```

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

4]

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

6 [-

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

8]

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

Na seqüência, bastaria realizar a substituição das 5 partes iguais (partes 1, 3, 5, 7, 9) por ‘F’:

1 F

2 [+

3 F

4]

5 F

6 [-

7 F

8]

9 F

o que resultaria: F[+F]F[-F]F

O mesmo resultado será obtido aplicando o pseudocódigo para a cadeia de caracteres do 3º. nível de iteração da gramática 1.

Estes procedimentos são possíveis devido à própria característica matemática das produções deste sistema. A cada nível de produção, o sistema injeta exponencialmente a mesma quantidade de caracteres, de acordo com a regra de produção estabelecida, em cada caractere ‘F’ presente na cadeia da regra. A solução para a determinação desta regra de uma cadeia, é justamente identificar o formato da gramática que a gerou, para então compactar cada cadeia que se repete em um ‘F’, a seqüência resultante deverá ser igual à regra de formação de uma determinada gramática.

4. Resultados

A figura 2 representa uma curva de Koch e a cadeia de caracteres que descreve esta figura é:

F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F-F-F+F+F+F-F-F+F+F+F-F-F+F+F+F-F-F+F+F+F-F-F+F-F+F-F-F+F+F+F-F-

Aplicando-se os procedimentos discutidos para se encontrar a regra geradora, têm-se:

1) Realizar a contagem de ‘F’: F_s=125

2) Encontrar o formato desta cadeia aplicando-se o pseudocódigo: o código retornará: F_r=5 e n=3

3) Executar um algoritmo de casamento de caracteres modificado para identificar F_r=5 partes iguais na cadeia:

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 +

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 -

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

6 -

7 F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F-F-F+F+F+F-F-F+F

8 +

9 F+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) Substituir as partes iguais por ‘F’:

1 F

2 +

3 F

4 -

5 F

6 -

7 F

8 +

9 F

5) Reagrupar as partes em forma linear e a solução está determinada:

Regra F+F-F-F+F. Esta solução é exatamente a regra amplamente conhecida para esta curva.

A figura 3 pode ser descrita pela seguinte cadeia de caracteres:

FF[+F][-F]FFF[+F][-F]F[+FF[+F][-F]F][-FF[+F][-F]F]FF[+F][-F]FFF[+F][-F]FFF[+F][-F]F[+FF[+F][-F]F][-FF[+F][-F]F]FF[

Aplicando-se os procedimentos discutidos para se encontrar a regra geradora, têm-se:

1) F_s=125

2) Pelo pseudocódico, será encontrado F_r=5 e n=3

3) Casamento de caracteres para identificar F_r=5 partes iguais na cadeia:

1 FF[+F][-F]FFF[+F][-F]F[+FF[+F][-F]F][-FF[+F][-F]F]FF[+F][-F]F

2 FF[+F][-F]FFF[+F][-F]F[+FF[+F][-F]F][-FF[+F][-F]F]FF[+F][-F]F

3 [+

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

5][-

6 FF[+F][-F]FFF[+F][-F]F[+FF[+F][-F]F][-FF[+F][-F]F]FF[+F][-F]F

7]

8 FF[+F][-F]FFF[+F][-F]F[+FF[+F][-F]F][-FF[+F][-F]F]FF[+F][-F]F

4) e 5) Substituindo as partes iguais por ‘F’, a regra resultante será: FF[+F][-F]F

Este resultado também coincide com os resultados encontrados nos livros em relação à Figura 3.

5. Considerações Finais

Encontrar a L-system que gere um determinado modelo não é uma tarefa fácil, uma vez que, dada uma cadeia de caracteres que descreve o modelo, não é simples encontrar uma gramática que a gere. Os procedimentos e métodos realizados apontam para uma solução inicial para gramáticas mais simples, em que o axioma é composto por apenas um símbolo com regra de produção variada. O próximo passo é melhorar a metodologia utilizada para a busca da base do logaritmo para que a busca se torne menos custosa e também resolva os casos de se ter outras possibilidades logarítmicas para uma mesma quantidade de caracteres ‘F’. Além disso, é necessário realizar estudos para gramáticas mais complexas (com mais de uma regra de produção) para que seja possível considerar um conjunto mais amplo de imagens iniciais.

Referências Bibliográficas

[1] Lindenmayer, A. Mathematical models for cellular interaction in development, J. Theoret. Biology, vol.18, pp.280-315, 1968.

[2] Prusinkiewicz, P. *at all.* The algorithmic beauty of plants. Springer-Verlag, 1990.

Anexos

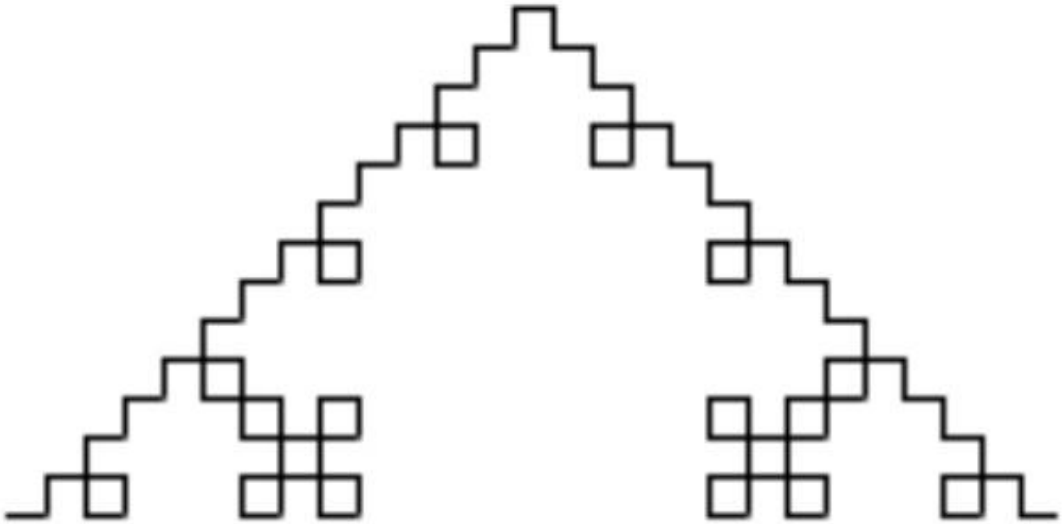

Figura 2