5ª MOSTRA ACADÊMICA UNIMEP
23 a 25 de outubro de 2007
UNIMEP Universidade Metodista de Piracicaba
Tema: EDUCAÇÃO BRASILEIRA: Extinção ou Sustentabilidade na Universidade

5º Congresso de Pós-Graduação

RESOLVENDO O PROBLEMA INVERSO DE LINDENMAYER PARA UMA SUBCLASSE DE GRAMÁTICAS L-SYSTEMS

Autor(es)

EDMAR SANTOS

Orientador(es)

REGINA CÉLIA COELHO

1. Introdução

O formalismo dos Sistemas de Lindenmayer, ou simplesmente L-Systems, foi introduzido pelo biólogo Aristid Lindenmayer em 1968 [1], baseado nas gramáticas de Chomsky [2], este sistema foi proposto primeiramente, para a representação dos processos de desenvolvimento de organismos vivos. Trata-se de um sistema gramatical de reescrita paralela e simultânea. As produções sofrem substituições paralelas em cada fase de desenvolvimento, todas ao mesmo tempo. Isto permite que tais classes de gramáticas sejam potencialmente recursivas, capazes de representar objetos complexos com muita facilidade. A representação formal das gramáticas L-systems é muito similar à definição das gramáticas de Chomsky. Esta representação pode ser feita por uma quádrupla ordenada {V, T, P, S} na qual V representa o conjunto dos símbolos variáveis sendo V um conjunto não vazio, normalmente são utilizados em V os símbolos {S, F}; T representa o conjunto dos símbolos terminais, geralmente são utilizados os símbolos {+,-, [,]}, podendo T ser um conjunto vazio. P representa o conjunto de regras de produção, estando P contido em VxV*. S representa o axioma, sendo S pertencente à V+. A principal diferença entre as gramáticas de Chomsky e as gramáticas de Lindenmayer, está na ordem de aplicação das regras de produção. Enquanto que nas gramáticas convencionais as regras de produção são aplicadas de forma seqüencial, nos L-Systems, a aplicação das regras ocorre de forma simultânea e paralela em toda a cadeia que será reescrita. As gramáticas L-Systems foram organizadas em duas grandes classes: as gramáticas L-Systems Livre de Contexto, ou OL-Systems, e as gramáticas L-Systems Sensíveis ao Contexto, ou IL-Systems. Os L-Systems são utilizados quase sempre associados às interpretações gráficas de suas produções. O mecanismo de interpretação gráfica, proposto por Prusinkiewicz [2], é baseado na geometria da tartaruga da linguagem Logo [37] e tornou-se o mecanismo clássico para as representações gráficas destes sistemas. Os L-Systems, tornaram-se alvo dos olhares científicos de muitos pesquisadores. A concepção original dos L-Systems era destinada à representação de processos de desenvolvimento de organismos vivos, com representações gráficas muito simples. Contudo, diante das potencialidades visualizadas nestes sistemas, outras abordagens mais aprimoradas foram apresentadas. Os L-Systems foram utilizados, por exemplo, na geração de fractais [3,6]; na síntese de imagens realísticas de árvores e plantas menores [5,6]; na análise do

processo de desenvolvimento de camadas celulares [7]; na geração de retas digitais [8]; no reconhecimento de formas vegetais em Visão Computacional [9,10]; na geração de notas musicais [11]; na procura de estruturas de redes neurais [12]; na síntese de criaturas virtuais para ambientes de simulação [13]; na síntese de neurônios artificiais [14,15,16,17], além muitas outras propostas. No entanto, o problema chave dos trabalhos que empregam o uso dos L-Systems, está exatamente em determinar qual o sistema adequado que represente convenientemente o processo de desenvolvimento do objeto de interesse. Este problema pode ser compreendido como um caso do Problema Inverso, teoria amplamente estudada em áreas como geofísica [18, 19], física [20], pesquisas em imagens médicas [21,22], além de outras. De forma geral, trata-se de um problema em que se pretende inferir a representação de um modelo fundamentado apenas na observação e análise de dados pertencentes ao mesmo sistema deste modelo [23,24,25,26]. Na teoria dos L-systems este problema é conhecido como o Problema Inverso de Lindenmayer. No campo da teoria das Linguagens Formais, a questão do Problema Inverso é conhecida como Inferência Gramatical, por vezes referenciada simplesmente por Indução Gramatical. Os processos de inferências gramaticais têm sido estudados, por exemplo, em áreas como aprendizagem de máquina [27], biologia computacional [28, 29, 30], reconhecimento de padrões em documentos estruturados [31], reconhecimento de fala [32]. O Problema Inverso de Lindenmayer aparece como uma particularidade da Inferência Gramatical, porém agora, no contexto do formalismo dos L-Systems [33]. Isto significa que, nos processos de inferências de gramáticas L-Systems, as propriedades destes sistemas devem ser consideradas. Cita-se o trabalho de KOZA [34] como um dos primeiros trabalhos nesta direção, apresentando uma metodologia para descobrir regras para os L-Systems baseada num algoritmo genético. KOZA apenas consegue obter uma regra similar, propondo ajustes manuais para torná-la idêntica à regra original. Em seus testes KOZA usou a gramática L-System construtora do fractal da Ilha Quadrática de Koch. Pode ser citado também, o trabalho apresentado por RUDOLPH e ALBER [35]. Trata-se de uma proposta para solução do problema Inverso de Lindenmayer no projeto de torres de transmissão. Sua proposta é fundamentada na execução de um algoritmo evolucionário que pretende convergir a geração da gramática no genótipo do objeto pretendido. O trabalho apresenta apenas a evolução dos objetos convergidos, sem, contudo, revelar as regras da gramática obtida em cada caso. Ainda outro trabalho pode ser referenciado, é o trabalho de COSTA e LANDRY [36]. Neste trabalho, é proposto formalmente um método para descrever fractais em forma de árvores e reconstruir estes modelos. Esta proposta consiste em reconstruir um objeto por meio de uma gramática gerada por um algoritmo genético, a partir de seqüências de imagens obtidas em torno do objeto original. Seus resultados apontam apenas para formas similares às amostras na maioria dos testes, além de que as regras da gramática encontrada utilizada na reconstituição do objeto não são apresentadas. Fica evidente, mesmo diante da contribuição destes trabalhos, que a questão do Problema Inverso dos L-Systems ainda desafia os pesquisadores.

2. Objetivos

Este trabalho pretende colaborar na solução de parte do Problema Inverso de Lindenmayer, direcionado para pesquisas na reconstrução de estruturas neurais em ambientes de realidade virtual. Para isso, é proposta uma metodologia que explora as características de auto-similaridade e reescrita paralela destes sistemas. A idéia consiste em reverter uma cadeia de caracteres, descritora de um determinado estado de desenvolvimento de uma estrutura auto-similar, em uma regra L-System. Esta regra deverá permitir a reprodução deste objeto novamente. Assume-se que estas cadeias já tenham sido obtidas por algum processo, seja manual ou automático. O Problema Inverso de Lindenmayer é visto como um problema de alta complexidade e não é tarefa simples resolvê-lo. Este trabalho explora nas classes de gramáticas L-Systems Livre de Contexto e Determinísticas, a subclasse de gramáticas que apresentem 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 neste trabalho foram obtidas a partir de uma gramática conhecida, conforme as restrições anteriores, porém, foram tratadas como sendo de origem desconhecida, a fim de permitir a validação da proposta. A intenção é apresentar uma metodologia que permita reencontrar a regra L-System original, utilizando para isso, somente uma cadeia de caracteres gerada por ela, independente do nível de produção em que esta cadeia foi 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 cadeia analisada.

3. Desenvolvimento

Conceitualmente, a idéia aqui apresentada é bastante intuitiva. Observe na figura 1, o esquema de produção de uma gramática L-System, que possui como regra de produção a seqüência “F[+F]F[-F]F”. Nesta figura, é possível notar que, a partir do primeiro nível de produção, todos os demais níveis são compostos exatamente pelas mesmas partes, ditadas pelo formato da regra de produção da gramática, pela presença dos símbolos não 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. Compreende-se então, que processo de reescrita dos L-Systems, confere às produções das gramáticas com um único símbolo no axioma e uma única regra de produção, um comportamento característico. Desta forma, é proposta a reversão das aplicações das produções na cadeia em questão. A idéia é reverter crescimento da cadeia promovido pelo processo de reescrita, de uma só vez, em todos os pontos de crescimento, como mostra o esquema de reversão, ainda na figura 1. Esta proposta de reversão pode ser organizada e expressa na forma do algoritmo apresentado na figura 2. Obviamente que a formalização destes conceitos requer a consideração de outros fatores. No processo de busca da regra geradora é necessário identificar estes pontos de evolução na cadeia. Da análise de cadeias de gramáticas conhecidas é possível estabelecer a relação entre a quantidade de ‘F’s presentes na cadeia e sua ordem de crescimento durante as produções. Esta relação está expressa na linha 5 do algoritmo apresentado na figura 2. Nesta relação, Ft representa a quantidade de caracteres ‘F’s contadas na cadeia, Fq representa a quantidade de ‘F’s presentes na regra de produção, e n representa o nível de produção da cadeia. Como somente Ft pode ser conhecido pela a contagem dos símbolos ‘F’s presentes na cadeia, é proposta uma busca iterativa em Fq para procurar valores para n. Uma vez que a iteração em Fq obtenha um valor inteiro para n, já que Ft não pode ser resultante de um expoente fracionário, já é possível conhecer o formato da regra de produção da cadeia analisada, isto é, saber quantos ‘F’s estarão presentes na regra de produção e qual o nível em que a cadeia analisada foi produzida. Na linha 7, é chamada a função Regressão() que recebe como parâmetros a cadeia e os valores de Fq e n. Esta função tem a finalidade de localizar e substituir por um único ‘F’ cada agrupamentos de caracteres que correspondem aos pontos de crescimento provocados pela regra de produção. Já na linha 8, aparece a função LimpaTerminais() que recebe como parâmetros a cadeia resultante da linha anterior e o valor de n. Algumas regras de produção provocam um acúmulo de símbolos não terminais antes e depois de cada ‘F’ espalhados pela cadeia. Esta função é chamada para retirar estes símbolos acumulados. Finalmente, na linha 9, a função ReproduzRegra() recebe a cadeia resultante da função anterior e realiza a produção desta cadeia até o valor de n. A reprodução desta regra, imediatamente após sua determinação, é importante por permitir o algoritmo ser encerrado, caso a regra encontrada tenha produzido uma cadeia exatamente igual à cadeia original fornecida na execução do algoritmo. Também é importante considerar que este algoritmo adota como solução, a menor regra encontrada que seja capaz de reproduzir a cadeia original. Já que outras regras maiores também podem ser responsáveis pela produção desta mesma cadeia. É o caso da cadeia “FFFFFFFFFFFFFFFF” que pode ter sido produzido pela aplicação de uma regra formada apenas por “FF”, em apenas quatro níveis de reescrita, ou pode ter sido gerada pela aplicação da regra “FFFF”, logo após o segundo nível de produção. Nestes casos, isto apenas implicaria em um processo de desenvolvimento mais vagaroso ou mais acelerado.

4. Resultados

Com base nas considerações anteriores, efetuou-se a análise de cadeias produzidas por várias gramáticas conhecidas, em níveis de produções diferentes. No item (a) da figura 3, é apresentada a cadeia w, para exemplificar, de forma resumida, a aplicação da metodologia proposta neste trabalho. As etapas necessárias para regressão da cadeia são apresentadas a seguir. Etapa 1: no item (b) da figura 3, aparece os pontos de crescimento identificados pelos sombreamentos claros e escuros; Etapa 2: no item (c) são mostrado

estes pontos de crescimento já substituídos por ‘F’s; Etapa 3: no item (d) aparecem identificados os caracteres terminais que devem ser removidos pela função LimpaTerminais(); Etapa 4: no item (e) é mostrada a regra L-System obtida a partir da cadeia w; A metodologia proposta também foi aplicada para cadeias produzidas por mais de 20 gramáticas, sendo algumas delas, variações de gramáticas mais conhecidas. Para cada gramática, foram geradas cadeias desde o nível um até o sexto nível de produção. No total, foram testadas mais de 120 cadeias diferentes. As principais observações sobre a aplicação da metodologia proposta sobre estas cadeias são: 1. O algoritmo proposto conseguiu obter a regra L-System de todas as cadeias submetidas ao teste; 2. O tempo de procura das regras L-Systems destas 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 aceitável. Por exemplo, na análise de uma das maiores cadeias testadas, contendo 1.328.601 caracteres (cerca de 245 páginas em um editor de texto comum) (cerca de páginas de um editor de texto comum), (cerca de páginas de um editor de texto comum),, a regra foi obtida em 28 minutos e 58 segundos; Todos os resultados apresentados foram obtidos em um sistema dotado com CPU de 1.800Mhz e 512MB de memória RAM.

5. Considerações Finais

Este trabalho apresentou uma proposta para solucionar parte do Problema Inverso de Lindenmayer em uma subclasse de gramáticas L-Systems. Os testes realizados até agora têm produzido resultados 100% exatos. Em todos os testes foi possível obter a regra de formação da cadeia de caracteres analisada, idêntica à regra original usada na síntese da amostra. Além disso, a execução do algoritmo na obtenção da regra tem apresentado um tempo de duração bastante aceitável. Trabalhos futuros envolvem o estudo de métodos para 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.

Referências Bibliográficas

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

[2] PRUSINKIEWICZ, P. Graphical applications of L-Systems. In: Proceedings of Graphics Interface 86 - Vision Interface 86, p. 247-253, 1986a.

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

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

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

[7] 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.

[8] BRONS, R. Theoretical and linguistic method for describing straight lines. In: EARN-SHAW, R. A. Algorithms for Computer Graphics, Spinger-Verlag, Berlin Heidel-berg, p. 19-57, 1995.

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

[10] SAMAL, A.; PETERSON, B.; HOLLIDAY, D. J. Recognition of plants using a stochas-tic L-System model. In: Journal of Electronic Imaging, SPIE, v. 11, part 1, p. 50-58, 2002.

[11] PRUSINKIEWICZ, P. Score generation with L-Systems. Proceedings of the 1986 International Computer Music Conference, p. 455-457, 1986b.

[12] 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.

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

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

[15] MCCORMICK, B. H.; MULCHANDANI, K. L-System modeling of neurons. Visualiza-tion in Biomedical Computing Conference Proceedings, 1994.

[16] 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.

[17] 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.

[18] MENKE, W. Geophysical data analysis: discrete inverse theory. Academic Press, 1989.

[19] PARKER, R. L. Geophysical inverse theory. Princeton University Press, 1994.

[20] BERTERO, M.; BOCCACCI, P. Introduction to inverse problem in imaging. Insti-tute of Physics Publishing, 1998.

[21] NATTERER, F.; WÛBBELING, F. Mathematical methods in image reconstruction. Society for Industrial

and Applied Mathematics, 2001.

[22] UHLMANN, G. Inside out: inverse problems and applications. Mathematical Sci-ences Research Institute Publications, 2003.

[23] CHALMOND, B. Modeling and inverse problems in image analysis. Springer, 2003.

[24] RAMM, A. G. Inverse Problems: mathematical and analytical techniques with appli-cations to engineering. Springer, 2005.

[25] TARANTOLA, A. Inverse problem theory and methods for model parameter. So-ciety for Industrial and Applied Mathematics, 2005.

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

[27] HIGUERA, C. de La. Current trends in grammatical inference. In: F.J.F. et al. Ad-vances in Pattern Recognition, Joint IAPR International Workshops SS-PR+SPR2000, Lecture Notes in Computer Science, Springer-Verlag, v. 1876, p. 28-31, 2000.

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

[29] 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.

[30] SAKAKIBARA, Y.; BROWN, M.; HUGHEY, R.; MIAN, I. S.; SJOLANDER, K.; UN-DERWOOD, R.; HAUSSLER, D. Stochastic context-free grammars for tRNA model-ing. Nuclear Acids Research. v. 22, p. 5112–5120, 1994.

[31] 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.

[32] 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.

[33] PRUSINKIEWICZ, P.; LINDENMAYER, A. The algorithmic beauty of plants. S-pringer-Verlag, New York, 1990.

[34] KOZA, J. R. Discovery of rewrite rules in Lindenmayer systems and state transition rules in cellular

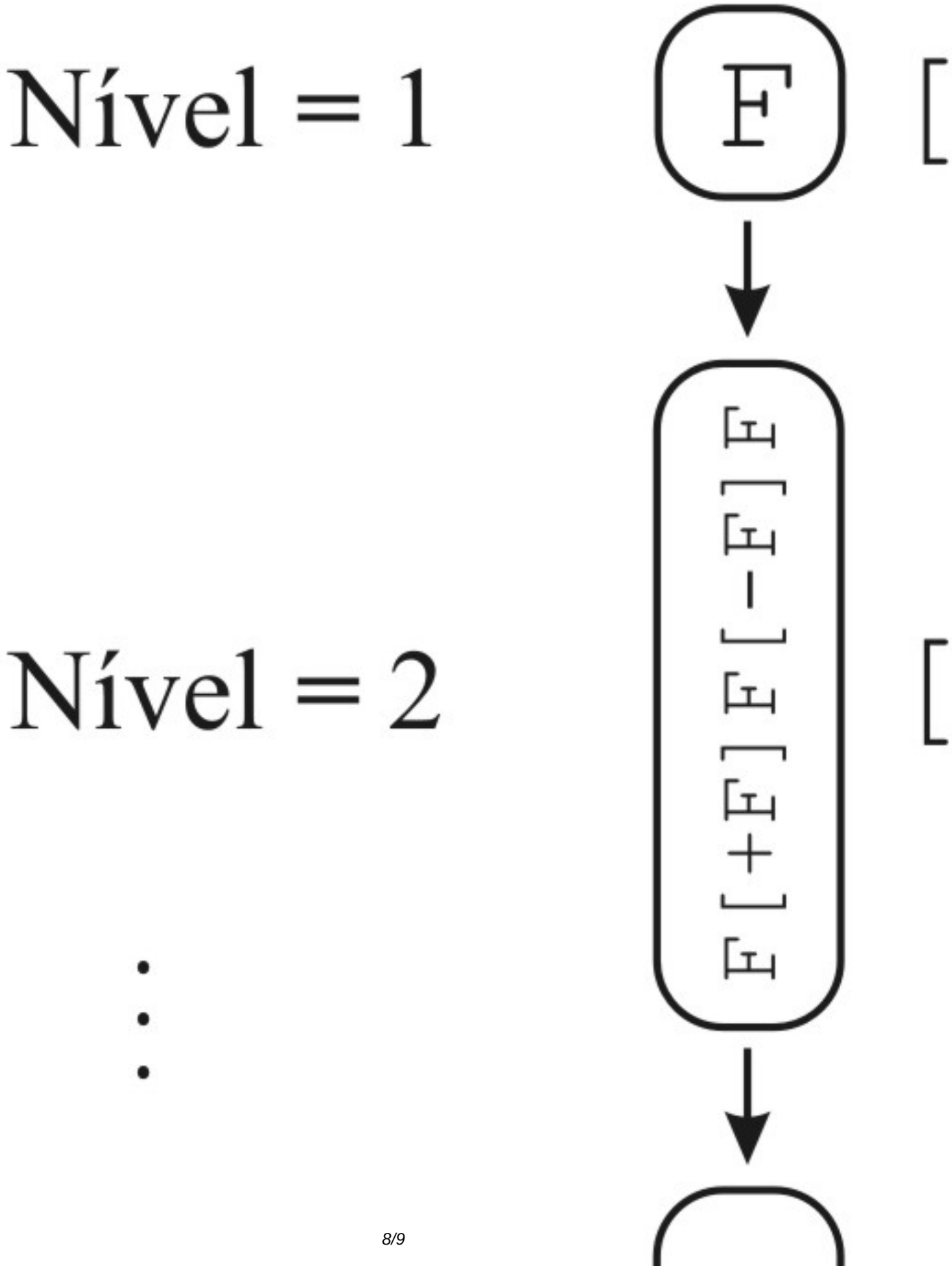
automata via genetic programming. Symposium on Pattern Forma-tion (SPF-93), Claremont, California, p. 1-19, 1993.

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

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

[37] PAPERT, S. Logo: computadores e educação. São Paulo: Brasiliense, 1986.

Anexos


Nível = 1
F
Nível = 2
F[+F]F[-F]F

Algoritmo

```
função INVERSEPROBLEM(w:string):string
1    Ft ← ContaSimbolo('F', w)
2    se Ft>1 faça
3        r ← w
4        para Fq ← 2 até int(raizquad(Ft)) faça
5            n ← ln(Ft)/ln(Fq)
6            se n for inteiro então
7                temp ← Regressao(w,Fq,n)
8                regra ← LimpaTerminais(temp,n)
9                producao ← Reproduz(regra,n)
10               se w = producao então
11                   r ← regra
12                   interromper para
13               fim se
14           fim se
15       fim para
16       retornar r
17   fim se
18   senão retornar w
```

Cadeia ***w*** :

a) ++++++F-F++F-F-++F-F++F-F++++F-F++F-F-++F-F++F-F-++++F-F++F
++F-F++F-F++++F-F++F-F-++F-F++F-F++++++F-F++F-F-++F-F++F-F+
F++F-F-++F-F++F-F-++++F-F++F-F-++F-F++F-F++++F-F++F-F-++F-F

b) ++++++F-F++F-F-++F-F++F-F++++F-F++F-F-++F-F++F-F-++++F-F++F
++F-F++F-F++++F-F++F-F-++F-F++F-F++++++F-F++F-F-++F-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) ++++++F-++++F++++++F-++++F

d) ++++++F-++++F++++++F-++++F

e) ++F-F++F-F