L-systems e o problema da parada

Lehilton Lelis Chaves Pedrosa,
José Carlos Loureiro Ralha (Orientador)[1]

[1] Departamento de Ciência da Computação
Universidade de Brasília

lehiltonchaves@yahoo.com.br,ralha@cic.unb.br

Abstract. *L-systems are parallel rewriting grammars used for several purposes, such as fractal description, automatic music composition, symbolic computation and biological modeling of plants or similar organisms. This work studies finiteness of the generated language. First, L-system convergence is defined and to illustrate this conception, a number sequence is ordered by means of grammar derivation. Then, an algorithm is presented to simulate Turing machines using L-systems. Finally, the halting problem is showed to be reducible to deciding whether an L-system's language is finite.*

Resumo. *Os L-systems são gramáticas com reescrita paralela e têm diversas aplicações, incluindo a descrição de fractais, composição automática de música e computação simbólica além da modelagem biológica de plantas e organismos similares. Esse trabalho estuda a finitude das linguagens associadas a L-systems. Primeiro, é definida a convergência de um L-system, exemplificada por uma gramática capaz de ordenar uma seqüência numérica. Em seguida, é apresentado um procedimento para simular máquinas de Turing com L-systems. Finalmente, demonstra-se que o problema da parada se reduz à tarefa de decidir se a linguagem de um L-system é finita.*

1. Introdução

O estudo das gramáticas com reescrita paralela, denominadas L-systems, iniciou-se com Lindenmayer como modelo matemático para interações entre células em desenvolvimento [Lindenmayer 1968]. Desde então, esse formalismo tem sido utilizado em diversas áreas, que vão desde aplicações gráficas, como a descrição de fractais por meio D0L-systems [Prusinkiewicz 1986a], à produção algorítmica de música [Prusinkiewicz 1986b].

Um L-system assemelha-se a uma gramática *Chomskyana* tradicional, mas diferencia-se fundamentalmente pelo fato de que os símbolos em uma cadeia são substituídos paralelamente — ou simultaneamente. Conseqüentemente, ao contrário das gramáticas tradicionais, nos L-systems, não há símbolos terminais e, portanto, a linguagem associada é o conjunto de todas as derivações [Salomaa 1973].

Os L-systems têm diversas variações, que incorporam noções de não-determinismo estocástico, determinismo, contexto à esquerda ou à direita, interpretação de suas derivações e gramáticas multidimensionais. Desse modo, os L-systems representam uma hierarquia de linguagem alternativa à de Chomsky [Lindenmayer and Rozenberg 1972]. As diferenças entre as gramáticas tradicionais para

as paralelas criam uma relação complexa entre as classes de linguagens comuns e os L-systems. Há, por exemplo, linguagem pode ser gerada por uma gramática sensível ao contexto do modelo de Chomsky, mas gerada por uma gramática livre de contexto na hierarquia dos L-systems [Salomaa 1973].

Dependendo da aplicação, diversas interpretações podem ser dadas aos símbolos do alfabeto considerado. Por exemplo, [Prusinkiewicz and Lindenmayer 1990] interpretam cada símbolo como um comando *Turtle-Graphics* para modelar o desenvolvimento de árvores e outras plantas, associando uma dimensão temporal discreta à seqüência de derivações de uma gramática. *Este trabalho interpreta o formalismo do sistema de reescrita como uma máquina teórica, cuja programação consiste em definir uma gramática L-system para um problema específico.*

O artigo está organizado da maneira a seguir. Primeiramente, uma revisão teórica sobre os L-systems é feita, abordando desde D0L-systems, que são livres de contexto determinísticos, até 2L-system paramétricos, sensíveis ao contexto. Em seguida, no intuito de definir um ponto de parada para uma derivação em uma gramática, é apresentada a noção de convergência de um L-system determinístico, quando nenhuma iteração do processo de reescrita altera a seqüência de derivações. Esse conceito é ilustrado com uma seqüência numérica específica, que será ordenada utilizando somente uma gramática paralela e, logo após, será demonstrada a corretude do modelo para uma seqüência qualquer. O resultado anterior é ampliado com um L-system que simula certa máquina de Turing, o que é depois generalizado com um algoritmo de conversão entre máquinas de Turing e L-systems — que é descrito neste trabalho — ilustrando o fato conhecido de que os L-systems podem simular máquinas de Turing arbitrárias [Lindenmayer and Rozenberg 1972].

Baseando-se no algoritmo apresentado, a condição de parada é investigada para o caso dos L-systems e, a partir dessa análise, demonstra-se que o problema da parada reduz-se à tarefa de decidir se a linguagem de uma gramática L-system é finita. Mostrar que não há tal procedimento é a principal contribuição deste trabalho.

2. L-systems

Os tipos de L-systems relevantes a este trabalho são descritos a seguir.

2.1. 0L-systems

Um 0L-system é um sistema de reescrita definido pela tripla $G = \langle V, \omega, P \rangle$, onde V é o *alfabeto*, i.e., um conjunto finito de símbolos, V^* é o conjunto das palavras construídas sobre esse alfabeto, $\omega \in V^+$ é a seqüência inicial de símbolos, denominada *axioma* e P é um conjunto finito de *produções*, tal que $P \subset V \times V^*$. Denotamos $(\alpha, \chi) \in P$ como $\alpha \rightarrow \chi$, em que α é chamado de *predecessor* e χ de *sucessor*.

Todo símbolo $\alpha \in V$ reescreve para uma seqüência $\chi \in V^*$ e, por convenção, se $\chi = \alpha$, não listamos tais produções em P. Se houver apenas uma produção para cada símbolo de V, então G é determinístico e é chamado de D0L-system.

Dado um 0L-system G, uma palavra $\nu = \chi_1 \dots \chi_n$ é derivada diretamente de outra $\mu = \alpha_1 \dots \alpha_n$, denotado $\mu \Rightarrow \nu$, se, e somente se, houver $\alpha_1 \rightarrow \chi_1, \dots, \alpha_n \rightarrow \chi_n$. Dizemos que uma palavra μ é uma *derivação* em G se ela é derivada diretamente de ω

ou se existirem $\mu_1, \ldots, \mu_n \in V^+$ tais que $\omega \Rightarrow \mu_1 \Rightarrow \mu_2 \Rightarrow \ldots \Rightarrow \mu_n$ e $\mu = \mu_n$, o que denotamos como $\omega \overset{*}{\Rightarrow} \mu$. Define-se o conjunto de todas as derivações em G como a *linguagem* de G, $L(G)$.

2.2. L-systems paramétricos

Por diversas vezes, para manter a definição de uma gramática compacta ou para representar modelos que dependam de constantes reais é conveniente associar um ou mais valores a cada símbolo de uma palavra. Esses valores são chamados de *parâmetros* dos símbolos e podem ser utilizados para impor condições sobre a derivação. Ao sistema de reescrita com essa extensão dá-se o nome de L-system paramétrico.

A cada símbolo é associado um conjunto ordenado de parâmetros reais, formando o que se denomina de *módulos*, M. As palavras são formadas por seqüências de módulos e o conjunto de todas elas é $M^* = (V \times \mathbb{R}^*)^*$, onde $\mathbb{R}^* = \mathbb{R}^0 \cup \mathbb{R}^1 \cup \mathbb{R}^2 \cup \ldots$ Os parâmetros são especificados nas produções por símbolos de Σ, denominados *parâmetros formais*, enquanto os valores associados aos símbolos são chamados de *parâmetros reais*. Denota-se $\mathcal{C}(\Sigma)$ o conjunto de todas as expressões lógicas de $\Sigma \cup \mathbb{R}$ e $\mathcal{E}(\Sigma)$ o conjunto de todas as expressões aritméticas.

Um 0L-system paramétrico é a tupla $G = \langle V, \Sigma, \omega, P \rangle$, em que $\omega \in M^+$ e $P \subset (V \times \Sigma^*) \times \mathcal{C}(\Sigma) \times (V \times \mathcal{E}(\Sigma))^*$ e uma produção $p \in P$ é escrita como

$$\alpha(a_1, a_2, \ldots, a_n) : C(a_1, a_2, \ldots, a_n) \rightarrow \prod_{i=0}^{m} \chi_i(E_i(a_1, a_2, \ldots, a_n), \ldots),$$

em que $\alpha, \chi_i \in V$, $a_1, a_2, \ldots, a_n \in \Sigma$, $n, m \geq 0$, $C \in \mathcal{C}(\Sigma)$ e $E_i \in \mathcal{E}(\Sigma)$. Quando C é uma tautologia, ela pode ser suprimida. Se $n = 0$, então recaímos sobre uma produção de um 0L-system tradicional e denota-se $\alpha \rightarrow \beta$, em que $\beta \in M^*$. Uma produção p se aplica a um módulo $\gamma(b_1, b_2, \ldots, b_t)$ de uma palavra paramétrica se $\alpha = \gamma$, $t = n$ e vale $C(b_1, b_2, \ldots, b_n)$.

Nos L-systems paramétricos a derivação é feita de forma tradicional, mas calculando os valores das expressões aritméticas $E_i(b_1, \ldots, b_n)$ para cada i e incluindo os resultados nos lugares de suas respectivas fórmulas.

2.3. 2L-systems paramétricos

Os 0L-systems referem-se à classe dos L-systems livres de contexto. Os 2L-systems são sensíveis ao contexto e contêm contextos opcionais à esquerda e à direita de cada predecessor estrito, α, de uma produção. Denota-se uma produção p como $\alpha_l < \alpha > \alpha_r \rightarrow \chi$ em que $\alpha_l, \alpha_r \in V$. Seja $w = \beta\alpha\gamma \in V^*$. A produção p se aplica ao símbolo α em w se, e somente se, existem β' e γ' tais que $\beta = \beta'\alpha_l$ e $\gamma = \alpha_r\gamma'$.

A extensão paramétrica para os 2L-systems é feita de maneira análoga aos sistemas livres de contexto e uma produção p tem a forma

$$\alpha_l(a_1,\ldots,a_l) < \alpha(a_{l+1},\ldots,a_m) > \alpha_r(a_{m+1},\ldots,a_n) : C(a_1,\ldots,a_n) \rightarrow \prod_{i=0}^{N} \chi_i(E_i(a_1, \ldots, a_n), \ldots),$$

em que $0 \leq l \leq m \leq n$ e p aplica-se a $w = ...\alpha_l(a_1, ..., a_s)\alpha(b_1, ..., b_t)\alpha_r(c_1, ..., c_u)$ se $s = l$, $t = m - l$, $u = n - m$ e vale $C(a_1, ..., a_s, b_1, ..., b_t, c_1, ..., c_u)$.

Este trabalho considera apenas L-systems determinísticos, daí, adicionalmente, exige-se que, para cada módulo $\alpha(a_1, ..., a_n)$, exista uma única produção para cada contexto e condição possíveis. Por notação, se o contexto ou a condição sobre os parâmetros forem indiferentes, esses são omitidos da especificação da gramática e, se não houver produção especificada, estará implícita a produção $\alpha(a_1, ..., a_n) \rightarrow \alpha(a_1, ..., a_n)$.

3. Máquina de Turing

De acordo com [Salomaa 1973], uma máquina de Turing é um sistema de reescrita $\langle V, F\rangle$ que tem as seguintes condições satisfeitas:

(i) O alfabeto, V, é particionado em *estados*, S, e *alfabeto da fita*, V_T.
(ii) Escolhem-se $s_0 \in S$, $\# \in V_T$ e o conjunto $S_1 \subset S$, respectivamente como, *estado inicial*, *marca de fronteira* e *estados finais*. Vale $V_1 = V_T \backslash \{\#\} \neq \emptyset$. Um elemento $\not b \in V_1$ e um subconjunto $V_I \subset V_1$ são especificados.
(iii) F é um conjunto de regras de produções da forma

$$s_i a \rightarrow s_j b \text{ (escrever na fita)}, \tag{1}$$
$$s_i ac \rightarrow a s_j c \text{ (mover para a direita)}, \tag{2}$$
$$s_i a\# \rightarrow a s_j \not b\, \# \text{ (mover para a direita e estender)}, \tag{3}$$
$$c s_i a \rightarrow s_j ca \text{ (mover para a esquerda), ou} \tag{4}$$
$$\# s_i a \rightarrow \# s_j \not b\, a \text{ (mover para a esquerda e estender)}, \tag{5}$$

em que $s_i, s_j \in S$ e $a, b, c \in V_1$. Para todos $s_i \in S$ e $a \in V_1$ existe no máximo uma produção da forma 1, 3 ou 5 tal que $s_i a$ é subpalavra do predecessor e, ainda, para cada $s_i, s_j \in S$ e $a \in V_1$, F não contém nenhuma produção da forma 2 e 3 (4 e 5), ou contém ambas para todo $c \in V_1$.

Adicionalmente, [Salomaa 1973] define $s\alpha$ palavra final, $s \in S$, $\alpha \in V_T$ se, e somente se, para todo a, tal que sa é subpalavra de um predecessor de F, α não começa com a. A linguagem da máquina de Turing é

$$L = \{\beta \in V_I^* : \#s_0\beta\# \Rightarrow^* \#\alpha s_1\gamma\#, s_1 \in S_1, \alpha, \gamma \in V_1^* \text{ e } s_1\gamma\# \text{ é final}\}.$$

4. Exemplo de L-system

O exemplo que segue utiliza um L-system como meio de realizar uma tarefa específica. Para isso, a entrada do problema é representada pelo axioma da gramática, que é derivado repetidamente, até se obter o resultado esperado. É importante ressaltar que, para os L-systems, não há uma palavra final definida naturalmente, uma vez que não existem símbolos terminais. Não obstante, se a definição de palavra final para máquina de Turing for adaptada para os L-systems, então pode-se determinar um ponto de parada para o procedimento de derivação.

Dizemos que uma palavra w é *final* se, dado um L-system determinístico, tem-se $w \Rightarrow w$. Nesse caso, diremos que um L-system *converge* para w quando $\omega \stackrel{*}{\Rightarrow} w$.

4.1. Ordenação baseada em L-system

Um problema clássico em Computação é a ordenação, em particular, de números. Partindo do princípio que uma seqüência pode ser ordenada apenas permutando elementos

vizinhos, pode-se definir tal permutação por uma reescrita de símbolos. A gramática (6) mostra que é possível ordenar uma certa seqüência com um 2L-system paramétrico.[1]

$$
\begin{array}{lccccccccccccc}
\omega: & mO(7)O(6)O(5)O(27)O(10)M & & & & & & & & & & & & \\
p_1: & m & < & O(n) & > & O(d) & : & & & n & \leq d & \rightarrow & O(n) \\
p_2: & m & < & O(n) & > & O(d) & : & & & n & > d & \rightarrow & O(d)O(n) \\
p_3: & O(e) & < & O(n) & > & O(d) & : & n \geq e & \text{e} & n & \leq d & \rightarrow & O(n) \\
p_4: & O(e) & < & O(n) & > & O(d) & : & n \geq e & \text{e} & n & > d & \rightarrow & O(d)O(n) \\
p_5: & O(e) & < & O(n) & > & O(d) & : & n < e & \text{e} & n & \leq d & \rightarrow & \varepsilon \\
p_6: & O(e) & < & O(n) & > & O(d) & : & n < e & \text{e} & n & > d & \rightarrow & O(d) \\
p_7: & O(e) & < & O(n) & > & M & : & n \geq e & & & & \rightarrow & O(n) \\
p_8: & O(e) & < & O(n) & > & M & : & n < e & & & & \rightarrow & \varepsilon
\end{array}
\tag{6}
$$

O L-system é formado pelos símbolos m, M e O. O módulo $O(n)$ representa um elemento n de uma seqüência numérica, m e M são delimitadores de início e fim. O axioma, $\omega = w_0$, e a palavra obtida depois de k derivações são da forma $w_k = mO(a_{k,1})O(a_{k,2}) \cdots O(a_{k,l})M$ e têm associadas seqüências $A_k = (a_{k,i})_{1 \leq i \leq l}$, com $l > 1$. Seja o axioma como em (6), daí

$$\omega = mO(7)O(6)O(5)O(27)O(10)M \quad \text{e} \quad A_0 = (7, 6, 5, 27, 10).$$

Após a primeira derivação, temos que m e M permanecem marcando o início e o fim da palavra, já que não há produção associada explicitamente. Para o módulo $O(7)$, $n = 7$ e as produções que se aplicam, pela análise do contexto, são p_1 e p_2, que tratam o caso do primeiro elemento. Se o módulo da direita tivesse o valor associado, d, com $n \leq d$, então o primeiro e o segundo elementos da seqüência A_0 estariam ordenados entre si e, por p_1, $O(n)$ seria mantido no lugar. Do contrário, como é o caso, $n = 7 > 6 = d$, o segundo módulo é inserido antes do primeiro, de forma que, na nova seqüência, A_1, $a_{1,1} = a_{0,2} = 6$ e $a_{1,2} = a_{0,1} = 7$. Na análise de contexto do módulo seguinte, aplicam-se p_3 a p_6. Como $a_{0,1} > a_{0,2}$, então o elemento $a_{0,2}$ já faz parte de A_1 e o módulo $O(n)$ é suprimido por p_5 ou p_6. Essa análise é repetida para os módulos seguintes, até o último, $O(10)$, quando a produção p_7 ou p_8 apenas o insere, ou o suprime, de acordo com o valor e associado ao módulo anterior. Esse processo é repetido até que as únicas produções aplicáveis sejam p_1, p_3 e p_7. As seqüências associadas às derivações são

$$A_1 = (6, 7, 5, 10, 27),\ A_2 = (6, 5, 7, 10, 27),\ A_3 = (5, 6, 7, 10, 27),\ A_4 = (5, 6, 7, 10, 27).$$

Para demonstrar a corretude da ordenação definida pelo L-system (6) com uma seqüência qualquer, deve-se mostrar que ele converge em t derivações e que $A_t = (a_{t,1}, a_{t,2}, \ldots, a_{t,l})$ com $a_{t,i} \leq a_{t,j} \forall i < j$. Por simplicidade de notação, denotaremos como $p_i(a)$ o sucessor do módulo a ao qual p_i se aplica.

Lema 1. *Seja $A = (a_i)_{1 \leq i \leq l}, l > 1$ uma seqüência parcialmente ordenada, tal que, para $i, j \geq k$, $a_i < a_j \Rightarrow i < j$ e $a_i \leq a_k$, $i < k$, então $w = m\ O(a_1) \cdots\ O(a_l)M \Rightarrow m\alpha\ O(a_k) \cdots\ O(a_l)\ M$, com $\alpha \in (\{O\} \times \mathbb{R})^*$ e $w' = m\ O(a_1) \cdots\ O(a_{k-1})\ M \Rightarrow m\alpha\ M$.*
Corolário. *Se $k = 1$, então A é ordenada e w é final.*

Dem. Aplicam-se às letras m e M as produções implicitas, $m \rightarrow m$ e $M \rightarrow M$, e, para o módulo $O(a_l)$, temos $n = a_l \geq a_{l-1} = e$, quando se aplica p_7. Para os módulos

[1] Em (6), os símbolos ω e $p_1, p_2, \ldots, p_8$ são usados como meio de referenciar o axioma e as regras. Esta convenção é usada neste e nos demais exemplos.

intermediários, $O(a_i)$, $i \geq k+1$, temos $n = a_i \geq a_{i-1} = e$ e $n = a_i \leq a_{i+1} = d$, caso em que p_3 se aplica. Defina q a produção que se aplica ao módulo $O(a_k)$. Se $k > 1$, então, novamente, $q = p_3$, do contrário, teremos $n = a_1 \leq a_2 = d$ e $q = p_1$. Portanto, $w \Rightarrow m\ p_{b_1}(O(a_1)) \cdots p_{b_{k-1}}(O(a_{k-1}))\ q(O(a_k))\ p_3(O(a_{k+1})) \cdots p_3(O(a_{l-1}))\ p_7(O(a_l))\ M = m\ p_{b_1}(O(a_1)) \cdots p_{b_{k-1}}(O(a_{k-1}))\ O(a_k) \cdots O(a_l)\ M$. Seja $w' \Rightarrow m\ p_{c_1}(O(a_1)) \cdots p_{c_{k-1}}(O(a_{k-1}))\ M$. Como a escolha da produção depende somente dos módulos à esquerda e à direita, então $b_i = c_i$ para $i < k-1$. Defina $r = p_{b_{k-1}}$ e $s = p_{c_{k-1}}$. Se $k-1 = 1$, então $r = p_1$, pois $n = a_{k-1} \leq a_k = d$ e s será a produção implícita $m < O(n) > M \rightarrow O(n)$. Se $k-1 > 1$, pela análise de contexto, s será p_7 ou p_8. Suponha $s = p_7$, então temos $a_{k-1} = n \geq e = a_{k-2}$ e, daí, r será p_3. Do mesmo modo, se $s = p_8$, $r = p_5$. Em qualquer caso, o sucessor de r é o mesmo de s. Portanto, $p_{c_1}(O(a_1)) \cdots p_{c_{k-1}}(O(a_{k-1})) = p_{b_1}(O(b_1)) \cdots p_{b_{k-1}}(O(a_{k-1}))$. □

Lema 2. *Seja* $w = \alpha O(a_1) \cdots O(a_n) M \Rightarrow \alpha'\ p_{b_1}(O(a_1)) \cdots p_{b_n}(O(a_n))\ M$, *com* $1 \leq b_i \leq 8$, *então* $\beta = p_{b_1}(O(a_1)) \cdots p_{b_n}(O(a_n))$ *com tamanho* $|\beta| \leq n$ *e, se* $\alpha = \alpha_e O(a_0)$, *com* $a_0 > a_1$, $|\beta| < n$.

Dem. Seja u o número de módulos $O(a_i), 1 \leq i \leq n$, com produções de sucessores de tamanho unitário, d os de sucessores duplos e v os de sucessores vazios, de tal modo que $n = u + d + v$. Seja j tal que $O(a_j)$ é o primeiro módulo cujo sucessor é duplo, então $p_{b_j} = p_2$ ou $p_{b_j} = p_4$, daí $a_j = n > d = a_{j+1}$. Para o módulo à direita, $O(a_{j+1})$, temos $n' = a_{j+1} < a_j = e'$, ou seja, esse módulo está fora de ordem com relação ao anterior, e a produção associada será ou p_5 ou p_8, que têm sucessores vazios, ou p_6, com sucessor unitário. Se a produção associada for p_6, então o próximo módulo também está fora de ordem e essa análise se repete, em uma seqüência de módulos com produções unitárias, até que não haja mais módulo O seguinte e p_8 se aplicará, ou algum módulo esteja ordenado com o anterior, quando p_5 se aplica. Isso significa que, para cada módulo com sucessor duplo, há um módulo com sucessor vazio posterior e antes do próximo com sucessor duplo. Portanto, $d \leq v$ e temos, $|\beta| = u + 2d \leq u + d + v = n$. Ainda, se $\alpha = \alpha_e O(a_0)$ e $a_0 > a_1$, analogamente, existe uma seqüência de módulos, a partir de $O(a_1)$, com sucessores unitários, seguidos de um módulo com sucessor vazio, que é anterior a $O(a_j)$, então, $d \leq v - 1 < v$ e temos, $|\beta| = u + 2d < u + d + v = n$. □

Lema 3. *Seja* $A = (a_1, a_2, \ldots, a_k, \ldots, a_l)$ *uma seqüência tal que* $l > 1$, $a_k \geq a_i, \forall i$ *e* $a_k > a_j, \forall j > k$, *então* $w = mO(a_1) \cdots O(a_l)M \Rightarrow^* m\alpha O(a_k)M$, *para algum* $\alpha \in (\{O\} \times \mathbb{R})^*$.

Dem. Por indução sobre $N = l - k$. Claramente, vale se $l = k$. Seja $A = (a_1, a_2, \ldots, a_k, \ldots, a_l)$, $N = l - k$ e suponha que a afirmação valha para $N' < N$. A palavra correspondente será $w = m\ O(a_1) \cdots O(a_{k-1})\ O(a_k)\ O(a_{k+1}) \cdots O(a_l)M$ e derivando teremos $w \Rightarrow m\ p_{b_1}(O(a_1)) \cdots p_{b_{k-1}}(O(a_{k-1}))\ p_{b_k}(O(a_k))\ p_{b_{k+1}}(O(a_{k-1})) \cdots p_{b_l}(O(a_l))M$, com $1 \leq b_i \leq 8$. Por um lado, se $k = 1$, então, pela análise de contexto do módulo $O(a_k)$, p_{b_k} é p_1 ou p_2 e, como $n = a_k > a_{k+1} = d$, $p_{b_k} = p_2$. Por outro lado, se $k > 1$, então p_{b_k} está entre p_3 e p_6. Como $n = a_k \geq a_{k-1} = e$ e $n = a_k > a_{k+1} = d$, $p_{b_k} = p_4$. Em qualquer caso o sucessor do módulo $O(a_k)$ será $O(a_{k+1})O(a_k)$. Defina $\beta = p_{b_{k+1}}(O(a_{k+1})) \cdots p_{b_l}(O(a_l))$ e $w' = m\ p_{b_1}(O(a_1)) \cdots p_{b_{k-1}}(O(a_{k-1}))\ O(a_{k+1})O(a_k)\beta M$, tal que $w \Rightarrow w'$. Pelo Lema 2, temos que $N' = |\beta| < l - k = N$ e, pela hipótese de indução, para a seqüencia A' associada a w', temos $w \Rightarrow w' \Rightarrow^* m\alpha O(a_k)M$, com $\alpha \in (\{O\} \times \mathbb{R})^*$. □

Tabela 1. Tabela de ações da máquina de Turing.

	Estado atual	Símbolo atual	Novo estado	Ação
r_1:	**0**	$\not b$	0	0
r_2:	0	0	1	$>$
r_3:	1	$\not b$	1	1
r_4:	1	1	0	$>$

Teorema 1. *1. O L-system (6) converge para w com seqüência associada A ordenada.*
2. w contém os mesmos elementos de ω.

Dem. Seja $w = mO(a_1)\cdots O(a_l)M \Rightarrow w'$.Todas as produções reinserem valores associados aos seus predecessores, a não ser p_5, p_6 e p_8. Suponha que uma dessas produções se aplique a um módulo $O(a_i)$, então existe um módulo $O(a_{i-1})$ à esquerda, para o qual vale $a_{i-1} = n > d = a_i$. A esse módulo se aplicará a produção p_2 ou p_4, que têm sucessores $O(a_i)O(a_{i-i})$, ou ainda p_6, com sucessor $O(a_i)$. Daí $|w'| \geq |w|$. Pelo Lema 2, temos $|w'| \leq |w|$. Portanto, $|w| = |w'|$. Resta provar o item 1. Por indução sobre $N = |\omega|$. Temos que a afirmação vale para $\omega = mM$, $N = 2$, e suponha que valha para $N' < N$. Seja $\omega = mO(a_1)\cdots O(a_l)M$, com $|\omega| = N$ e A_0 a seqüência associada. Pelo Lema 3, $\omega \Rightarrow^* m\alpha O(a_k)M$, tal que $a_k \geq a_i, \forall i$. Considere um segundo L-system igual ao do enunciado, mas substitua o axioma por $\omega' = m\alpha M$. Temos $|\omega'| = N - 1 < N$, então, pela hipótese de indução, $\omega' \Rightarrow^* w' = m\alpha' M$ e a seqüência associada a w', $A' = (a'_1, \ldots, a'_{l-1})$, é ordenada. Pelo Lema 1, $\omega \Rightarrow^* m\alpha O(a_k)M \Rightarrow^* m\alpha' O(a_k)M = w$, com seqüência associada $A = (a'_1, \ldots, a'_{l-1}, a_k)$. Como $a_k \geq a'_i, \forall i$, A é ordenada e, pelo Corolário do Lema 1, w é final. □

5. Simulação de Máquina de Turing baseada em L-system

Pode-se ampliar o poder computacional de L-systems, ilustrado na seção anterior, pela simulação de outros modelos de computação. Considere uma máquina de Turing que, a cada passo, ou escreve na fita um símbolo do alfabeto ou move-se para a direita, de acordo com a tabela 1, cujo estado inicial é 0 e a configuração de partida da fita é toda preenchida com brancos, $\not b$. Essa máquina escreve a seqüência $010101\ldots$ na fita.

Considere o L-system (7) onde $\#define\ X\ n$ significa que X é uma constante numérica de valor n. Seja o mapeamento $\{0 \leftrightarrow S_0, 1 \leftrightarrow S_1, \not b \leftrightarrow S_{\not b}\}$, do alfabeto da

$$
\begin{array}{llllllll}
\#define\ S_0\ 0 & & \#define\ S_1\ 1 & & \#define\ S_{\not b}\ 2 & & & \\
\omega: & LH(S_{\not b},0)R & & & & & & \\
p_1: & & H(c,s) & : & c = S_{\not b} \text{ e } s = 0 & \rightarrow & H(S_0,0) & \\
p_2: & H(c,s) < & S(r) & : & c = S_0 \text{ e } s = 0 & \rightarrow & S(c)H(r,1) & \\
p_3: & H(c,s) < & R & : & c = S_0 \text{ e } s = 0 & \rightarrow & S(c)H(S_{\not b},1)R & \\
p_4: & & H(c,s) & : & c = S_0 \text{ e } s = 0 & \rightarrow & \varepsilon & \\
p_5: & & H(c,s) & : & c = S_{\not b} \text{ e } s = 1 & \rightarrow & H(S_1,1) & \\
p_6: & H(c,s) < & S(r) & : & c = S_1 \text{ e } s = 1 & \rightarrow & S(c)H(r,0) & \\
p_7: & H(c,s) < & R & : & c = S_1 \text{ e } s = 1 & \rightarrow & S(c)H(S_{\not b},0)R & \\
p_8: & & H(c,s) & : & c = S_1 \text{ e } s = 1 & \rightarrow & \varepsilon &
\end{array}
\tag{7}
$$

máquina de Turing na tabela 1 às constantes definidas para o L-system. O estado, σ_n, a configuração da fita, C_n, após n passos e a palavra, w_n, após n derivações são:

n	σ_n	C_n	w_n
0	0	$\ldots \not b \mathbf{\not b} \not b \ \ldots$	$LH(S_{\not b}, 0)R$
1	0	$\ldots \not b \mathbf{0} \not b \ \ldots$	$LH(S_0, 0)R$
2	1	$\ldots \not b 0 \mathbf{\not b} \not b \ \ldots$	$LS(S_0)H(S_{\not b}, 1)R$
3	1	$\ldots \not b 0 \mathbf{1} \not b \ \ldots$	$LS(S_0)H(S_1, 1)R$
4	0	$\ldots \not b 01 \mathbf{\not b} \not b \ \ldots$	$LS(S_0)S(S_1)H(S_{\not b}, 0)R$
5	0	$\ldots \not b 01 \mathbf{0} \not b \ \ldots$	$LS(S_0)S(S_1)H(S_0, 0)R$
6	1	$\ldots \not b 010 \mathbf{\not b} \not b \ \ldots$	$LS(S_0)S(S_1)S(S_0)H(S_{\not b}, 1)R$
$\vdots$		$\vdots$	$\vdots$

Observando a seqüência de derivações e comparando com a seqüência de configurações da máquina de Turing, nota-se que existe uma correspondência. O axioma, ω, está relacionado com a configuração inicial da fita, C_0, e assim por diante. O mapeamento entre uma configuração da fita e uma palavra do L-system pode ser feito associando: o lado esquerdo da fita, $\ldots \not b$, a L; o direito, $\not b \ldots$, a R; 0 a $S(S_0)$; 1 a $S(S_1)$; e o símbolo i na posição de leitura do cabeçote, $i = 0, 1$, com o estado da máquina, σ, a $H(S_i, \sigma)$. Além disso, as regras da tabela relacionam-se com as produções: r_1 com p_1; r_2 com p_2, p_3 e p_4; r_3 com p_5; e r_4 com p_6, p_7 e p_8.

O algoritmo na figura 1 generaliza o exemplo dado e servirá de base para o resultado seguinte. Ele recebe como entrada o alfabeto da máquina de Turing, V, com $\not b \ \in V$, a configuração inicial, C_0, e um conjunto de regras $F \subset \{(s_0, a_0, s, a) : s_0, s \in S, a_0 \in V, a \in V \cup \{<, >\}, s_0 \neq s$ ou $a_0 \neq a\}$, em que $S \subset \mathbb{N}$ é o conjunto de estados e, se $a \in \{<, >\}$, então o cabeçote se move para a esquerda ou para direita e, caso contrário, o cabeçote escreve o símbolo a na posição atual da fita. A saída do algoritmo é um axioma, $\omega = f(C_0)$, e um conjunto de produções, P, de um 2L-system paramétrico correspondente.

Lema 4. *Seja M uma máquina de Turing e G o L-system obtido pelo algoritmo na figura 1 com configuração inicial C_0. Logo, M pára se, e somente se, G converge.*

Dem. Sem perda de generalidade, podemos supor $s_0 = 0$ e $S \subset \mathbb{N}_0$. Seja $M = \langle V, F \rangle$ uma máquina de Turing e $G = \langle V', \omega, P \rangle$ o L-system correspondente gerado pelo algoritmo. Considere $C = \#a_1 \ldots a_{l-1} s_i a_l \ldots a_n \#$, com $s_i \in S, a_k \in V_1, 1 \leq k \leq n$ e $w = f(C) = LS(S_{a_1}) \ldots S(S_{a_{l-1}})H(S_{a_l}, s_i)S(S_{a_{l+1}}) \ldots S(S_{a_n})R$. Suponha que exista $r \in V$ que se aplique a C, então, $C \Rightarrow C_1$, $w \Rightarrow w_1$ e $w_1 = f(C_1)$, pois, por exemplo, para r da forma $s_i a_l \rightarrow s_j b$, $C \Rightarrow C_1 = \#a_1 \ldots a_{l-1} s_j b a_{l+1} \ldots a_n \#$ e, pelo passo 3.c do algoritmo, existe $p \in P$, $H(c, s) : c = S_{a_l}$ e $s = s_i \rightarrow H(S_b, s_j)$, e, daí, $w \Rightarrow LS(S_{a_1}) \ldots S(S_{a_{l-1}})\ H(S_b, s_j)\ S(S_{a_{l+1}}) \ldots S(S_{a_n})R = f(C_1)$. Para r de outra forma, a verificação é análoga. Portanto, se $C_0 \Rightarrow^t C_t$, então $\omega \Rightarrow^t w_t = f(C_t)$. Por um lado, se M pára, ou seja, se não existe C_{t+1} com $C_t \Rightarrow C_{t+1}$, então nenhuma produção de G tem sua condição satisfeita para w_t, caso em que as produções implícitas $H(c, s) \rightarrow H(c, s)$ e $S(s) \rightarrow S(s)$ se aplicam e, então, $w_t \Rightarrow w_t$. Por outro lado, se G é convergente, então existe $f(C_t) = w_t \Rightarrow w_{t+1} = w_t$, com $C_0 \Rightarrow^t C_t$. Suponha que M não pára, então existe $r \in V$ que se aplica a C_t e $C_t \Rightarrow C_{t+1} = f(w_{t+1}) = f(w_t) = C_t$. Daí r só pode ser da forma $s_0 a_0 \rightarrow sa$, com $s_0 = s$ e $a_0 = a$, mas temos, pelas restrições impostas sobre a entrada do algoritmo, que $s_0 \neq s$ ou $a_0 \neq a$. Contradição. □

1. Defina $i \leftarrow 0$ e $P \leftarrow \emptyset$.
2. Enquanto houver $w \in V$, faça $V \leftarrow V \setminus \{w\}$, $i \leftarrow i + 1$
   e defina $S_w \leftarrow i$.
3. Enquanto houver $r = (s_0, a_0, s_1, a_1) \in F$, faça $F \leftarrow F \setminus \{r\}$ e,
   (a) se a_0 for $<$,
   $P \leftarrow P \cup \{S(e) > H(c,s) : c = S_{a_0}$ e $s = s_0 \rightarrow H(e, s_1)S(S_{a_0}),$
   $L > H(c,s) : c = S_{a_0}$ e $s = s_0 \rightarrow LH(S_{\not b}, s_1)S(S_{a_0}),$
   $H(c,s) : c = S_{a_0}$ e $s = s_0 \rightarrow \varepsilon \}$;
   (b) mas, se a_1 for $>$,
   $P \leftarrow P \cup \{H(c,s) < S(d) : c = S_{a_0}$ e $s = s_0 \rightarrow S(S_{a_0})H(d, s_1),$
   $H(c,s) < R : c = S_{a_0}$ e $s = s_0 \rightarrow S(S_{a_0})H(S_{\not b}, s_1)R,$
   $H(c,s) : c = S_{a_0}$ e $s = s_0 \rightarrow \varepsilon\}$;
   (c) senão,
   $P \leftarrow P \cup \{H(c,s) : c = S_{a_0}$ e $s = s_0 \rightarrow H(a_1, s_1)\}$.
4. Faça $a \leftarrow \not b$ ou encontre o primeiro símbolo a diferente de $\not b$ de C_0, se houver,
   e defina $\omega \leftarrow LH(S_a, 0)$.
5. Enquanto houver símbolo diferente de $\not b$ ainda não escolhido de C_0,
   escolha o próximo símbolo a de C_0 e faça $\omega \leftarrow \omega S(S_a)$.
6. Faça $\omega \leftarrow \omega R$ e termine.

Figura 1. Algoritmo de conversão de máquina de Turing para L-system

Lema 5. *Seja G um 2L-system paramétrico qualquer. Se existe algoritmo P que decida se $L(G)$ é finita, então existe um algoritmo C que decide se G converge.*

Dem. Se $P(G)$ for "não", então defina $C(G) =$ não. Do contrário, existe t tal que $\omega \Rightarrow^t w \Rightarrow w$, ou $\omega \Rightarrow^t \omega$. Faça $\omega = w_0 \Rightarrow w_1 \Rightarrow \cdots \Rightarrow w_{t+1}$. Se existir i, com $w_i \Rightarrow w_i$, então defina $C(G) =$ sim, senão, defina $C(G) =$ não. □

Teorema 2. *Dado um 2L-system paramétrico G, não existe algoritmo que decida se $L(G)$ é finita.*

Dem. Suponha que exista algum procedimento P tal que, dada a entrada G, forneça a saída $P(G) =$ sim, se G é finita, ou $P(G) =$ não, caso contrário. Seja C como no Lema anterior, M uma máquina de Turing e C_0 sua configuração inicial. O algoritmo na figura 1, diga-se A, fornece a saída $G = A(M, C_0)$. Defina Q o procedimento de saída $Q(M, C_0) = C(A(M, C_0))$. Suponha que M pare com configuração inicial C_0, então $G = A(M, C_0)$ converge e $Q(M, C_0) =$ sim; caso contrário, G não converge e $Q(M, C_0) =$ não. Portanto, existe um procedimento Q que decide o problema da parada, o que é um absurdo. □

6. Conclusão

Os L-systems têm uso prático para modelagem em diversas áreas, dependendo do significado que se dá aos símbolos de seu alfabeto. O conhecimento da linguagem associada à gramática aplica-se, portanto, ao modelo representado, que pode ser, por exemplo, o desenho de uma planta. Em particular, é importante decidir se a linguagem é limitada ou cresce indefinidamente.

Foi apresentado um método de classificação baseado em L-systems e demonstrou-se sua corretude, pelo teorema 1, utilizando-se o argumento de que a linguagem gerada

pela gramática do modelo converge e, por conseguinte, é finita. Esse exemplo não só mostra a importância de se conhecer o tamanho da linguagem, como comprova o potencial dos L-systems de resolver certos tipos de problema, aproveitando-se do paralelismo da gramática.

Embora fosse desejável a existência de um procedimento automático que decidisse se a linguagem de um L-system é finita, o teorema 2 responde negativamente. Para verificar isso, foi utilizado um algoritmo que fornece um L-system capaz de simular uma máquina de Turing arbitrária e, com base nessa simulação, demonstrou-se que o problema da parada seria redutível a um tal procedimento, caso houvesse.

Para simplificar o algoritmo, ele foi restrito a máquinas que não contenham regras que deixem sua configuração inalterada, do tipo $s_i a \rightarrow s_i a$. Isso porque, se não fosse feita a restrição, o algoritmo poderia fornecer um L-system convergente para uma máquina de Turing que não pára. Essa restrição pode ser suprimida, embora não altere o resultado do teorema 2. Para isso, basta especificar um caso para essas produções, na figura 1, que forneça a produção $H(c, s) : c = S_a$ e $s_i = s_i \rightarrow H(a_i, s_i)\xi$ acompanhada de outra, $\xi \rightarrow \xi\xi$, para evitar que o L-system convirja, onde ξ não representa nenhuma célula da fita.

Este trabalho esteve limitado a 2L-systems paramétricos e, a princípio, o enunciado do teorema 2 não poderia se aplicar aos 2L-systems. No entanto, como o número de símbolos e estados de uma máquina de Turing é finita e não foi utilizada nenhuma fórmula diferente do próprio parâmetro nos sucessores das produções, pode-se interpretar os parâmetros como partes dos símbolos, que satisfazem às condições paramétricas apenas pela análise de contexto, formando 2L-systems tradicionais. Desse modo, o enunciado se aplica também aos 2L-systems. Em trabalhos futuros, a existência de um procedimento que decida se a linguagem de L-systems de classes mais restritas são finitas pode ser investigada, particularmente de D0L-systems.

Agradecimentos

Agradeço ao Prof. Guilherme A. Pinto por ter questionado sobre a relação entre o problema da parada e os L-systems. Esse questionamento levou ao trabalho aqui apresentado.

Referências

Lindenmayer, A. (1968). Mathematical Models for Cellular Interactions in Development. *Journal of Theoretical Biology*, 18:280–315.

Lindenmayer, A. and Rozenberg, G. (1972). Developmental systems and languages. In *STOC '72: Proceedings of the fourth annual ACM symposium on Theory of computing*, pages 214–221, New York, NY, USA. ACM Press.

Prusinkiewicz, P. (1986a). Graphical applications of L-systems. In *Proceedings on Graphics Interface '86/Vision Interface '86*, pages 247–253, Toronto, Canada.

Prusinkiewicz, P. (1986b). Score Generation with L-Systems. *Proceedings of International Computer Music Conference*, pages 455–457.

Prusinkiewicz, P. and Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Springer, New York.

Salomaa, A. (1973). *Formal Languages*. Academic Press, New York.