Lindenmayer system

Canonical name	LindenmayerSystem
Date of creation	2013-03-22 18:57:31
Last modified on	2013-03-22 18:57:31
Owner	CWoo (3771)
Last modified by	CWoo (3771)
Numerical id	11
Author	CWoo (3771)
Entry type	Definition
Classification	msc 92C80
Classification	msc 68Q45
Synonym	L system
Synonym	DL-system
Synonym	PL-system
Defines	L-system
Defines	start word
Defines	deterministic L-system
Defines	propagating L-system
Defines	L-language
Defines	EL-system
Defines	2L-system
Defines	1L-system

Definition

Lindenmayer systems, or L-systems for short, are a variant of general rewriting systems. Like a rewriting system, an L-system is also a language generator, where words are generated by applications of finite numbers of rewriting steps to some initial word given in advance. However, unlike a rewriting system, rewriting occurs in *parallel* in an L-system. The notion of L-system was introduced by plant biologist Aristid Lindenmayer when he was studying the growth development of red algae.

Formally, an *L-system* is a triple $G = (\Sigma, P, w)$, where

1. Σ is an alphabet,

2. w is a word over Σ, and

3. P is a finite subset of $\Sigma \times \Sigma^*$ such that for every $a \in \Sigma$, there is at least one $u \in \Sigma^*$ such that $(a, u) \in P$.

w is called the *start word*, or the *axiom* of G, and elements of P are called productions of the L-system G, and are written $a \to u$ instead of (a, u).

As stated above, an L-system is a language generator, where words are generated from the axiom w by repeated applications of productions of P. Let us see how this is done. Define a binary relation $\Rightarrow$ on Σ^* as follows: for words $u, v \in \Sigma^*$,

> $u \Rightarrow v$ iff either $u = a_1 \cdots a_n$ and $v = v_1 \cdots v_n$, where $a_i \to v_i \in P$, or $u = v$.

Now, take the transitive closure $\Rightarrow^*$ of $\Rightarrow$ and set

$$L(G) := \{u \mid w \Rightarrow^* u\}.$$

Then $L(G)$ is called the *language generated* by the L-system G. An L-language is $L(G)$ for some L-system G.

Examples

1. Let $G = (\{a\}, \{a \to a^2\}, a)$. In two derivations, we get $a \Rightarrow a^2 \Rightarrow a^2a^2 = a^4$. It is easy to see that after n derivations, we get $a \Rightarrow^* a^{2^n}$, and that $L(G) = \{a^{2^n} \mid n \geq 1\}$. Note that if parallel rewriting is not required then a^3 may be derived in three steps: $a \Rightarrow a^2 \Rightarrow (a^2)a = a^3$.

2. Let $G = (\{a\}, \{a \to a, a \to a^2\}, a)$. Then $L(G) = \{a\}^+$.

3. Let $G = (\{a\}, \{a \to \lambda, a \to a^2\}, a)$. Then $L(G) = \{a^{2n} \mid n \geq 0\} \cup \{a\}$.

4. Let $G = (\{a, b\}, \{a \to ab, b \to ba\}, a)$. Then we get a sequence of words $a \Rightarrow ab \Rightarrow abba \Rightarrow abbabaab \Rightarrow \cdots$, and $L(G)$ is the set containing words in the sequence. Note the recursive nature of the sequence: if u_n is the nth word in the sequence, then $u_1 = a$ and $u_{n+1} = u_n h(u_n)$, where h is the homomorphism given by $h(a) = b$ and $h(b) = a$.

5. L-systems can be used to generate graphs. Usually, symbols in Σ represent instructions on how to construct the graph. For example,

$$G = (\{a, b, c\}, \{a \to a, b \to b, c \to cacbbcac\}, c)$$

generates the famous Koch curve. If $u \in L(G)$ is derived from c in n steps, then u represents the n-th iteration of the Koch curve. To draw the n-th iteration based on u, we do the following:

   (a) write $u = d_1 \cdots d_m$, where $d_i \in \Sigma$ (it is easy to see that $m = 2^{n-1}$).

   (b) at each d_i, a current position z_i, and current direction θ_i, are given.

   (c) start at the origin on the Euclidean plane in the positive x direction, so that $z_0 = (0, 0)$ and $\theta_0 = 0$.

   (d) upon reading d_i, where $i > 0$:

      - if $d_i = a$, set $z_i = z_{i-1}$ and $\theta_i = \theta_{i-1} + 60$,
      - if $d_i = b$, set $z_i = z_{i-1}$ and $\theta_i = \theta_{i-1} - 60$,
      - if $d_i = c$, draw a line segment of unit length from z_{i-1} to a point P based on θ_{i-1}, and set $z_i = P$ and $\theta_i = \theta_{i-1}$.

A production $b \to u$ is said to correspond to $a \in \Sigma$ if $b = a$. Both productions in Example 2 correspond to a. A production is said to be a constant production if it has the form $a \to a$. A symbol in Σ is called a *constant* if the only corresponding production is the constant production. In the last example above, a, b are both constants. a is not a constant in Example 2, even though it has a corresponding constant production.

Properties

Given an L-system $G = (\Sigma, P, w)$, we can associate a function $f_G : \Sigma \to 2^{\Sigma^*}$ as follows: for each $a \in \Sigma$, set

$$f_G(a) := \{u \mid a \to u \in P\}.$$

Then f_G extends to a substitution $s_G : \Sigma^* \to 2^{\Sigma^*}$. It is easy to see that $s_G(w)$ is just the set of words derivable from w in one step: $s_G(w) = \{u \mid w \Rightarrow u\}$. In fact,

$$L(G) = \bigcup \{s_G^n(w) \mid n \geq 0\},$$

where $s_G^0(w) = \{w\}$, and $s_G^{n+1}(w) = s_G(s_G^n(w))$.

In relation to languages described by the Chomsky hierarchy, we have the following results:

1. Every L-language is context-free.
2. If an L-system $G = (\Sigma, P, w)$ contains a constant production for each symbol in Σ, then $L(G)$ is context-free.
3. Denote the families of regular, context-free, context-sensitive, and L-languages by $\mathscr{R}, \mathscr{F}, \mathscr{S}, \mathscr{L}$, and set $\mathscr{X}_1 = \mathscr{R}$, $\mathscr{X}_2 = \mathscr{F} - \mathscr{R}$, and $\mathscr{X}_2 = \mathscr{S} - \mathscr{F}$. Then $\mathscr{L} \cap \mathscr{X}_i$ and $\overline{\mathscr{L}} \cap \mathscr{X}_i$ are non-empty for $i = 1, 2, 3$. Here, $\overline{\mathscr{L}}$ is the complement of $\mathscr{L}$ in 2^{Σ^*}, the family of all languages over Σ.

Subsystems

An L-system is said to be *deterministic* if every symbol in Σ has at most one (hence exactly one) production corresponding to it. A deterministic L-system is also called a DL-system. Examples 1,4,5 above are DL-systems. For a DL-system, the associated substitution is a homomorphism, which means that for each $n \geq 0$, the set $s_G^n(w)$ is a singleton, so we get a unique sequence of words $w_0, w_1, \ldots$, such that $w_n \Rightarrow w_{n+1}$. If $|w_n| < |w_{n+1}|$ for some n, then the word sequence is infinite. In particular, if w_n is a prefix of w_{n+1} for all large enough n, and the lengths of the words have the property that $|w_n| = |w_{n+1}|$ implies $|w_m| < |w_{m+1}|$ for some $m > n$, then the DL-system defines a unique infinite word (by taking the union of all finite words). In Example 4 above, the infinite word we obtain is the famous Thue-Morse sequence (an infinite word is an infinite sequence).

An L-system is said to be *propagating* if no productions are of the form $a \to \lambda$. A propagating L-system is also called a PL-system. All examples above, except 3, are propagating. A DPL-system is a deterministic propagating L-system. In a DPL-system, the lengths of the words in the corresponding sequence are non-decreasing, and one may classify DPL-systems by how fast these lengths grow.

Variations

There are also ways one can extend the generative capacity of an L-system by generalizing some or all of the criteria defining an L-system. Below are some:

1. Create a partition of $\Sigma = N \cup T$, the set N of non-terminals and the set T of terminals, so that only terminal words are allowed in $L(G)$. Such a system is called an EL-system. Formally, an EL-system is a 4-tuple

   $$H = (N, T, P, w)$$

   such that $G_H = (N \cup T, P, w)$ is an L-system, and $L(H) = L(G_H) \cap T^*$.

2. Notice that the productions in an L-system are context-free in the sense that during a rewriting step, the rewriting of a symbol does not depend on the "context" of the symbol (its neighboring symbols). This is the reason why an L-system is also known as a 0L-system. We can generalize an 0L-system by permitting context-sensitivity in the productions. If the rewriting of a symbol depends both on its left and right neighboring symbols, the resulting system is called a 2L-system. On the other hand, a 1L-system is a system such that dependency is one-sided.

   Formally, a 2L-system is a quadruple

   $$(\Sigma, P, w, \sqcup).$$

   Both Σ and w are defined as in an L-system. $\sqcup$ is a symbol not in Σ, denoting a blank space. P is a subset of $\Sigma_1 \times \Sigma \times \Sigma_1 \times \Sigma^*$, where $\Sigma_1 = \Sigma \cup \{\sqcup\}$, such that for every $(a, b, c) \in \Sigma_1 \times \Sigma \times \Sigma_1$, there is a $u \in \Sigma^*$ such that $(a, b, c, u) \in P$. Elements of P are called productions, and are written $abc \to u$ instead of (a, b, c, u). Rewriting works as follows: the binary relation $\Rightarrow$ on Σ^* called a rewriting step, is given by $u \Rightarrow v$ iff either $u = v$, or $u = a_1 \cdots a_n$ and $v = v_1 \cdots v_n$, such that

(a) $\sqcup a_1 a_2 \to v_1$,

(b) $a_{i-1} a_i a_{i+1} \to v_i$ where $i = 2, \cdots, n-1$, and

(c) $a_{n-1} a_n \sqcup \to v_n$.

If $n = 2$, then productions of the second form above do not apply. If $n = 1$, then $\sqcup a_1 \sqcup \to v_1$ are the only productions.

A 1L-system is then a 2L-system such that either, $abc \to u$ for some $c \in \Sigma_1$ implies $abd \to u$ for all $d \in \Sigma_1$, or $cab \to u$ for some $c \in \Sigma_1$ implies $dab \to u$ for all $d \in \Sigma_1$.

It is easy to see that an L-system is a 2L-system such that if $abc \to u$ for some $a, c \in \Sigma_1$, then $dbe \to u$ for all $d, e \in \Sigma_1$.

3. Allow the possibility that not all of the symbols may be rewritten. This means that $u \Rightarrow v$ iff either $u = v$, or $u = a_1 \cdots a_n$ and $v = v_1 \cdots v_n$, and either $a_i = v_i$ or $a_i \to v_i \in P$.

4. Allow more than one axiom. In other words, the single axiom word w is replaced by a set W of axioms.

References

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