Equivalencias entre algunos sistemas complejos: Fractales, Autómatas Celulares y Sistemas de Lindenmayer

Alfonso Ortega de La Puente.

Julio del año 2000.

Salmos 23 (Vg22) 1-6

Para Carlos García-Diego.

Este trabajo, entre otras cosas, no habría sido imaginable sin él.

Tampoco sin la presencia, confianza y esfuerzo del tutor Manuel Alfonseca Moreno, sin mis padres, sin Piolín, Obélix y el resto de mi familia, sin la paciencia y el intercambio de ideas con Marina, Cayetano y con Juan.

Yo sólo he tenido que escribir.

Índice General

Parte I

Introducción

Capítulo 1

Introducción, objetivos y plan de la tesis

1.1 Introducción y objetivos

La potencia expresiva, propiedades y aplicaciones de los lenguajes formales de la jerarquía de Chomsky y de las máquinas abstractas relacionadas con ellos están incluidas en los planes de los estudios universitarios de informática desde hace muchos años. Estos conocimientos permiten racionalizar la construcción de dispositivos electrónicos digitales, estudiar los problemas que se pueden resolver con ellos, y formalizar la manera de expresarlos. La Informática Teórica resulta, por tanto, una materia básica para la construcción de sistemas informáticos (tanto en la programación de su *software* como en el diseño y construcción de su *hardware*) y para el estudio teórico de la capacidad y potencia de los mismos. Pero también constituye un punto de apoyo imprescindible para abordar los objetivos de la inteligencia artificial, una de las áreas de la informática en la que mayor esfuerzo en investigación se está realizando en los últimos años. La manipulación de sistemas formales (como son las gramáticas de Chomsky y las arquitecturas abstractas relacionadas con ellas) se sigue considerando como uno de los prerrequisitos para comportamientos que simulen algún aspecto de inteligencia. Líneas de actuación como el lenguaje natural, la traducción automática, el razonamiento automático basado en la lógica formal, el almacenamiento, en bases de conocimiento, y manipulación, mediante sistemas expertos, de conocimento con formalismos simbólicos, el incremento de la potencia deductiva de los gestores de bases de datos, la implementación de tareas de minería de datos y de sistemas de ayuda a la decisión, el aprendizaje automático, etc. ... son líneas de interés y actuación de la inteligencia artificial que están en la actualidad en un grado de desarrollo inimaginable sin el fundamento de la teoría de lenguajes formales, gramáticas y autómatas.

La simulación mediante ordenador de complejos procesos físicos en cualquier rama científica ha sido siempre un área de interés debido a su versatilidad, economía y posibilidad de reproducir experimentos cuya realización podría resultar costosa e incluso imposible.

Durante las últimas décadas se está reactivando la investigación sobre arquitecturas abstractas y lenguajes formales distintos de los asociados a la jerarquía de Chomsky. La profun-

dización en el estudio de las redes de neuronas artificiales, autómatas celulares y otras estructuras matemáticas discretas ha mostrado la complejidad de su naturaleza (tanto las redes neuronales, como los autómatas celulares, por ejemplo, son sistemas dinámicos de comportamiento complejo). Su potencia expresiva es grande. Se demuestra que sencillas configuraciones de algunas de estas estructuras son computacionalmente completas y por tanto capaces de expresar y resolver los mismos problemas que cualquier ordenador (entendido como aproximación física a la más potente de las máquinas abstractas relacionadas con la jerarquía de Chomsky: la máquina de Turing).

Muchas de estas arquitecturas pueden tratar la información de forma simbólica. Este hecho ha motivado la inclusión de la simulación entre sus aplicaciones: si se es capaz de describir de forma simbólica un proceso concreto y existe una arquitectura abstracta asociada a esa descripción simbólica, se habrá diseñado un simulador para ese proceso. Es el caso, por ejemplo, de los autómatas celulares. Otras formalizaciones de este tipo han sido construidas con el objetivo explícito de simular algunos procesos biológicos. Es el caso, por ejemplo, de los sistemas de Lindenmayer. Las características de las estructuras abstractas y formales facilitan que su manipulación resulte más flexible o intuitiva que la del proceso real. Este enfoque supone un nuevo punto de vista para afrontar la simulación, ya que utiliza un "aparato matemático" distinto y más directo que la expresión de la variación en el tiempo de las variables descriptoras o que la generación de un modelo estadístico de comportamiento similar; y un "soporte físico" más flexible y potente: la naturaleza formal de esta aproximación la hacen implementable sobre cualquier sistema informático y esto resulta más flexible y potente que la construcción de un dispositivo electrónico analógico dedicado a comportarse igual que el proceso simulado. Simulación digital contínua, simulación estadística y simulación analógica han sido los enfoques tradicionales a los que se viene a sumar la simulación mediante arquitecturas formales discretas.

La geometría ha experimentado lo que se puede considerar una revolución en las últimas tres décadas. La realidad no resulta ser geométricamente monstruosa. Esta sensación paradójica: "la naturaleza nos rodea, la vemos, la tocamos, nos alimentamos de ella pero no podemos describirla formalmente porque no se ajusta a la geometría que somos capaces de entender", se ha mantenido desde Euclides hasta nuestros días. La definición de los fractales ha tendido al fin un puente entre la realidad y la geometría. Encontramos las características de los objetos fractales en multitud de formas de la naturaleza. Y no sólo eso. Puede identificarse las mismas propiedades en el comportamiento de muchos sistemas físicos, de hecho los fractales son capaces de explicar la complejidad de la dinámica de muchos sistemas que estaban siendo estudiados y simulados en distintas disciplinas. Los fractales resultan tener un enorme poder expresivo para describir la complejidad.

El primer objetivo de la presente tesis es continuar el estudio de la expresividad de los sistemas L y buscar relaciones formales con otras maneras de expresar fenómenos y procesos complejos. En particular, se ha comenzado con algunas características sencillas de los autómatas celulares y de los fractales.

Las relaciones entre sistemas L y fractales han sido ampliamente estudiadas de una manera más bien sintomática. El innegable atractivo visual de las imágenes fractales generadas por ordenador hace frecuente encontrar multitud de programas que los generan y en ellos se suele añadir la posibilidad de describir fractales según Lindenmayer. El usuario ve que las gramáticas

que ha escrito *generan algunos fractales*. Realmente ve que una traducción gráfica de cada símbolo de las cadenas derivadas por las gramáticas que escribe representa una curva que recuerda poderosamente a su percepción de algunos fractales. A medida que consigue cadenas más largas (derivaciones de profundidad mayor) la interpretación gráfica le presenta una aproximación mejor de la curva fractal. Como *se ve* que es la misma curva *se sabe* que el sistema L representa al fractal.

El objetivo de la presente tesis es estudiar formalmente esta relación con la intención de analizar propiedades de los fractales a través del estudio de los sistemas de Lindenmayer que los representan.

Las relaciones entre los sistemas L y los autómatas celulares existen desde el mismo origen de los sistemas L. Ambos tienen como objetivo la descripción de la conducta de sistemas complejos tales como el comportamiento de los organismos pluricelulares. Formalizar correctamente el crecimento y desarrollo de estos organismos era el objetivo de los sistemas L. Entre otras razones, Lindenmayer definió estos sistemas porque los autómatas celulares, al igual que otros formalismos, le parecían más adecuados para expresar la conducta de los organismos que su crecimiento. Sin embargo, el paralelismo entre los autómatas celulares y los sistemas de Lindenmayer casi no ha sido estudiado desde entonces. Esta tesis representa uno de los primeros esfuerzos en esa línea.

El segundo objetivo de la presente tesis es abordar formalmente las relaciones entre autómatas celulares y sistemas L.

1.2 Plan de la tesis

La tesis está compuesta por seis partes: la primera de ellas es introductoria, en la segunda y en la tercera se presentan los resultados obtenidos en el uso de sistemas de Lindenmayer para el estudio respectivamente de fractales y de autómatas celulares. La cuarta parte enumera las conclusiones y las líneas abiertas. Las dos últimas partes contienen los apéndices y las referencias bibliográficas.

Cada parte está divida en capítulos . A continuación se resume el contenido de las cuatro primeras partes.

Parte I, presentación

Esta parte está formada por los cinco primeros capítulos.

En el primer capítulo se introduce la motivación, los objetivos y la estructura de latesis.

En el segundo capítulo se repasan los conceptos de Informática Teórica necesarios para la comprensión de la tesis.

Los siguientes tres capítulos introducen y definen los conceptos utilizados por la tesis, el segundo está dedicado a los sistemas de Lindenmayer, el tercero a los autómatas celulares y el cuarto a los fractales.

Parte II, estudio de fractales mediante sistemas de Lindenamyer

Esta parte contiene tres capítulos: los que llevan los números seis, siete y ocho.

En el capítulo seis se define el concepto de gráfico de Lindenmayer. Los gráficos de Lindenmayer se utilizan para representar fractales del tipo *iniciador-iterador*. Se realiza una clasificación de estos gráficos de forma que se identifica varios subconjuntos intersantes, en particular los gráficos de Lindenmayer tortuga invariantes al ángulo, los gráficos de Lindenmayer vectoriales relacionados racionalmente y los gráficos de Lindenmayer tortuga con estructura simple.

En el capítulo siete se demuestra que los dos subconjuntos de gráficos de Lindenmayer del párrafo anterior son equivalentes.

En el capítulo ocho se define la dimensión total de Lindenmayer de un gráfico de Lindenmayer con estructura simple, que se calcula estudiando las cadenas derivadas por el sistema de Lindenmayer del gráfico. Se identifica y analiza el problema de la precisión y la posibilidad de que la interpretación gráfica de las cadenas utilizadas para el cálculo de la dimensión sea una curva con solapamientos. Como consecuencia de este análisis se define la representación canónica de un punto del plano $\Re^2$ accesible mediante un gráfico tortuga de Lindenmayer y la dimensión efectiva de Lindenmayer de un gráfico de Lindenmayer con estructura simple. Para finalizar el capítulo se analiza las relaciones entre las dos dimensiones definidas.

Parte III, estudio de autómatas celulares mediante sistemas de Lindenamyer

Esta parte está formada por los capítulos noveno y décimo.

En el capítulo noveno se analiza tres ejemplos de autómatas celulares no probabilistas y se obtiene el sistema de Lindenmayer que se comporta como ellos.

El capítulo décimo comienza con el análisis del comportamiento de los autómatas celulares probabilistas. También se define y analiza el comportamiento de los sistemas L n-dimensionales probabilistas. Posteriormente se aborda la equivalencia entre ellos y para ello se define el la relación de equivalencia *paso a paso* y se demuestra que para cualquier autómata celular n-dimensional probabilista puede construirse un sistema L n-dimensional probabilista equivalente paso a paso a él. El capítulo termina con un ejemplo y con una conclusión acerca del posible uso del resultado demostrado en casos más sencillos como, por ejemplo, los del capítulo anterior.

Capítulo 2

Repaso de Informática Teórica

El estudio de los lenguajes formales, las máquinas que tienen asociadas y los problemas que se pueden expresar y resolver con ellos son el objetivo de la disciplina que se conoce como Informática Teórica.

La Informática Teórica, como otras muchas ramas de la informática, tiene un origen y carácter marcadamente interdisciplinares. Su desarrollo se ha fundamentado en los avances de campos científicos aparentemente no relacionados entre sí como son, entre otros, la lingüística, la teoría de máquinas y los fundamentos de las matemáticas. Sus aplicaciones han permitido reconocer y formalizar ciertos fenómenos similares en muy diferentes disciplinas.

A continuación se presenta una breve cronología de las áreas de conocimiento más relevantes en la génesis de la Informática Teórica y sus relaciones.

2.1 Formalización, algoritmo: nacimiento de la informática teórica y de la teoría de la computabilidad

Los trabajos de David Hilbert se pueden citar como antecedentes de este campo científico. David Hilbert se propuso encontrar un método general para decidir si una fórmula lógica era verdadera o falsa. Éste es el enunciado del problema de la decisión o *Entscheidungproblem* que en 1900 se propuso resolver. Una línea de pensamiento que ha encontrado en su camino paradojas y limitaciones pero ha permitido formalizar la lógica matemática y sentar las bases de la automatización del razonamiento.

El más severo revés a las pretensiones de encontrar un sistema completo y consistente para la demostración de cualquier teorema se produjo años después. Los trabajos de Kurt Gödel (por ejemplo "*On formally undecidable propositions in Principia Mathematica and related systems*" de 1930) mostraron las limitaciones de estos sistemas al enunciar lo que se podría resumir de la siguiente forma: "Toda formulación axiomática consistente de la teoría de números contiene proposiciones indecidibles". Dicho de otro modo, cualquier teoría matemática es incompleta.

Los trabajos de Gödel fueron continuados por, entre otros, Alan Mathison Turing (1912-1953) en obras como "*On computable numbers with an application to the Entscheidungproblem*", de

1937 en el que desarrolla el teorema de Gödel.

Turing logra formalizar la idea de algoritmo. El concepto de algoritmo se atribuye al matemático "Abu Ja'far Mohammed ibn Musa al - Jowârizmî". En un principio se entendía por algoritmo cualquier conjunto de reglas que permitiera obtener un resultado determinado a partir de ciertos datos de partida. Turing formalizó esta idea. Definió una máquina abstracta que lleva su nombre, que era capaz de adoptar una infinidad de posibles configuraciones y que sirvió para definir formalmente los conceptos de algoritmo y de problema computable: algoritmo era todo aquello que podía "entender" (ejecutar) la máquina de Turing y todo problema o tarea que pudiera ser resuelto por la máquina de Turing (expresable mediante un algoritmo) era computable. La infinidad de posibles estados hace que la construcción física de la máquina de Turing resulte imposible. Los ordenadores, sin embargo, pueden entenderse como una aproximación bastante fiel.

Fruto de esta formalización fue la identificación de ciertos problemas que no podían ser expresados mediante algoritmos y que, por tanto, no podrían ser resueltos ni por la máquina de Turing ni por ningún ordenador. Los trabajos de Turing se pueden considerar como el origen tanto de la Informática Teórica como de la Teoría de la Computabilidad.

2.1.1 Breve cronología de autómatas y máquinas secuenciales

A finales de los años 30, en los trabajos de Claude Elwood Shanon (como por ejemplo "*A symbolic analysis of relay and switching circuits*", de 1938) los circuitos combinatorios y secuenciales fueron aplicados a la lógica matemática. Estos trabajos supusieron el inicio de lo que, tras el desarrolló de las décadas siguientes, se conoce como teoría de las máquinas secuenciales y de los autómatas finitos.

El concepto de autómata se restringió desde la idea general e informal de un sistema cualquiera capaz de transmitir información hasta la definición y clasificación de arquitecturas abstractas con diferentes características que les confieren diferentes propiedades estudiadas con detalle y precisión.

Desde entonces, las aplicaciones de los autómatas son diversas y amplias. Son utilizados en áreas muy dispares en las que se identifica como característica común el tratamiento símbolico de información. Algunos campos de aplicación son las comunicaciones (teoría de comunicación, redes de conmutadoras y codificadores), teoría de control de procesos, diseño de ordenadores y lenguajes de programación (lógica de los circuitos secuenciales, estructura y análisis de los lenguajes de programación para ordenadores digitales, teoría algebraica de lenguajes), aplicaciones propias de inteligencia artificial (reconocimiento de patrones, fisiología del sistema nervioso, traducción automática de lenguajes), vida artificial (teoría lógica de los sistemas evolutivos y autoreproductivos), etc.

Alfabetos, símbolos y estados

Los objetos matemáticos básicos en los autómatas son el *símbolo* o *letra* (unidad de información que el autómata puede intercambiar con el exterior), los *estados* (descripciones de las situaciones en las que se puede encontrar) y el alfaberto (conjunto finito no vacío de símbolos o de estados)

Entre todos los tipos de autómatas se mencionará los siguientes

Autómatas finitos deterministas

Intuitivamente, un autómata finito determinista es un dispositivo capaz de recibir información del exterior en forma de símbolos de un alfabeto de entrada y transitar a otros estados dependiendo del símbolo que haya recibido y del estado en el que se encuentre. Una misma combinación de estado y símbolo del alfabeto de entrada produce una única transición siempre al mismo estado.

Se llama *autómata finito determinista* a la quíntupla

$$(\Sigma, Q, f, q_0, F)$$

Donde

- Σ es un alfabeto llamado *alfabeto de entrada.*
- Q es otro alfabeto llamado *conjunto de estados.*
- $f : Q \times \Sigma \to Q$ se llama *función de transición* y determina el estado al que transita el autómata cuando recibe como entrada un símbolo del alfabeto de entrada.
- $q_0 \in Q$ es el *estado inicial.*
- $F \subset Q$, $F \neq \Phi$ es el *conjunto de estados finales* o *estados de aceptación.*

Autómatas finitos no deterministas

Intuitivamente un autómata finito no determinista es un dispositivo capaz de recibir información del exterior en forma de símbolos de un alfabeto de entrada y transitar a otros estados dependiendo del símbolo que haya recibido y del estado en el que se encuentre. Una misma combinación de estado y símbolo del alfabeto de entrada puede producir transiciones a diferentes estados.

Se llama *autómata finito no determinista* a la séxtupla

$$(\Sigma, Q, f, q_0, F, T)$$

Donde

- Σ, Q, q_0 y F han sido definidos previamente.
- $f : Q \times \Sigma \to \wp(Q)$ determina un conjunto de estados a los que puede transitar un autómata cuando se encuentar en un estado determinado y recibe un símbolo del alfabeto de entrada.
- $T \in Q \times Q$ se utiliza en el caso de que el autómata permita transiciones entre estados sin recibir ninguna entrada.

Autómatas finitos probabilistas

Intuitivamente, un autómata finito probabilista es un dispositivo que generaliza el no determinismo de las transiciones al especificar qué probabilidad tiene cada transición de producirse.

Se llama *autómata finito probabilista* a la quíntupla

$$(\Sigma, Q, M, P(0), F)$$

Donde

- Σ, Q y F han sido definidos previamente.
- M es un conjunto de $\#(Q)$ *matrices de dimensión* $\#(Q) \times \#(Q)$ *de probabilidad de transición* o *estocásticas de transición* que especifican las probabilidades de transitar de un estado a otro.
- $P(0) \in ([0,1] \cap \Re)^{\#Q}$ es el *vector de estado inicial* que especifica la probabilidad de que inicialmente el autómata esté en cada uno de los posibles estados.

Autómas a pila

Intuitivamente, un autómata a pila es un dispositivo que tiene acceso a los símbolos de una cinta de entrada y al símbolo superior de una unidad de memoria con estructura de pila. El dispositivo puede encontrarse en un estado determinado y en cada momento transita a otro estado en función de su estado anterior, el símbolo de entrada y el símbolo de la pila de memoria. El dispositivo también podría modificar el estado de la pila sin necesidad de leer símbolo alguno de la cadena. Estos autómatas son no deterministas porque en cada paso de su funcionamiento pueden elegir entre un conjunto de posibles acciones.

Formalmente, un *autómata a pila* es una séptupla

$$(\Sigma, \Gamma, Q, A_0, q_0, f, F)$$

Donde

- Σ es el *alfabeto de entrada,*
- Γ es el *alfabeto de la pila,*
- Q es un *conjunto finito de estados,*
- $A_0 \in \Gamma$ es el *símbolo inicial de la pila,*
- $q_0 \in Q$ es el *estado inicial del autómata,*
- $f : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to \wp(Q \times \Gamma^*)$ determina las *transiciones* del autómata como se ha comentado previamente de manera informal.
- $F \subset Q$ es el *conjunto de estados finales* que no necesariamente es propio.

Máquinas de Turing

Intuitivamente, una máquina de Turing es un dispositivo que dispone de una cabeza de lectura y escritura en una cinta. Esta cabeza puede escribir o leer un símbolo cada vez. La máquina puede adoptar como estado un elemento cualquiera de Q. En cada instante, y en función del símbolo que lea de la cinta, la máquina realiza las tres acciones siguientes:

- Transita a un nuevo estado.
- Escribe un símbolo en la cinta en la posición que acaba de leer.
- La máquina se detiene o bien mueve la cabeza de lectura y escritura una posición a la izquierda o a la derecha.

Formalmente, se llama *máquina de Turing* a la séptupla

$$(\Gamma, \Sigma, b, Q, q_0, f, F)$$

Donde

- Γ es el *alfabeto de símbolos de la cinta.*
- $\Sigma \subset \Gamma$ es el *alfabeto de símbolos de entrada.*
- $b \in \Gamma - \Sigma$ es el *símbolo blanco.*
- Q, q_0, y F han sido definidos previamente.
- $f : Q \times \Gamma \to Q \times \Gamma \times \{I, D, P\}$, donde $\{I, D, P\}$ es el conjunto de direcciones, I indica izquierda, D indica derecha y P indica parada. Es una correspondencia que determina el siguiente estado, el símbolo que se escribirá y la dirección (o parada) de movimiento como respuesta al estado actual y al símbolo de la cadena recién leído.

Máquinas de Turing no deterministas

Puede ser necesario especificar varias posibles acciones para una máquina de Turing como respuesta a la misma situación , sin aportar información más precisa acerca de cómo es elegida cada una de ellas.

Se llama *máquina de Turing no determinista* a la la séptupla

$$(\Gamma, \Sigma, b, Q, q_0, f, F)$$

Donde

- $\Gamma, \Sigma, b, Q, q_0, F$ han sido definidos previamente al describir la máquina de Turing.
- $f : Q \times \Gamma \to \wp(Q \times \Gamma \times \{I, D, P\})$, es decir, una correspondencia que, en lugar de asignar un único elemento de $Q \times \Gamma \times \{I, D, P\}$ a cada combinación de $Q \times \Gamma$ le asigna un subconjunto de $Q \times \Gamma \times \{I, D, P\}$ que indica que puede realizar indistintamente cualquiera de las acciones representadas por sus elementos.

Autómatas lineales acotados

Se llama *autómata lineal acotado* a una máquina de Turing no determinista cuya cinta tiene longitud limitada por ambos extremos. Como consecuencia de esta característica deja de ser necesario el símbolo blanco b y es necesario incorporar a Σ dos símbolos delimitadores que indican el principio y el final de la cadena. Ejemplo de estos símbolos son los siguientes:

- $\vdash$ para representar el comienzo o extremo izquierdo.
- $\dashv$ para representar el final o extremo derecho.

Estos símbolos indican los límites que no puede traspasar la cabeza de lectura y escritura: no podrá ir a la derecha del extremo derecho ni a la izquierda del extremo izquierdo.

Formalmente un autómata lineal acotado es la óctupla

$$(\Gamma, \Sigma \cup \{\vdash, \dashv\}, \vdash, \dashv, Q, q_0, f, F)$$

Donde todos los símbolos han sido descritos formal o informalmente en los párrafos precedentes.

2.1.2 Introducción a los lenguajes formales

La figura más relevante en el área de los lenguajes formales es Noam Chomsky. En algunos de sus trabajos publicados entre las décadas de los 50 y 70 ("*Three models for the description of language*" (1956), "*Syntactic structures*" (1957), "*On certain formal properties of grammars*" (1959), "*Context-free grammars and push-down storage*" (1962), "*Formal properties of grammars*" (1963), "*The algebraic theory of context-free languages*" (1963), "*Aspectos de la Teoría de la Sintaxis*" (1972)) estableció las bases de la disciplina tal y como se conoce en la actualidad.

Símbolos, alfabetos, palabras

Los objetos matemáticos básicos en la teoría de lenguajes formales son el *símbolo* o *letra* (mediante cuya concatenación se formarán palabras) y el *alfabeto*, conjunto finito no vacío de símbolos.

Se llama *palabra formada con los símbolos de un alfabeto* a toda secuencia finita (tupla sin longitud determinada) de símbolos de ese alfabeto. Las palabras se representarán mediante la concatenación de sus símbolos en el mismo orden en el que aparecen en la palabra.

Es necesario definir la palabra que consiste en 0 símbolos del alfabeto considerado (puede ser definida sobre cualquier alfabeto) que se llama *palabra vacía* y se representa con la letra griega *lambda* (λ).

Sean dos palabras x, y formadas con los símbolos del mismo alfabeto Σ, llamaremos *concatenación de las palabras x e y*, y a esta operación se le asignará el símbolo del punto "." (o simplemente xy cuando el contexto deje claro que se está haciendo referencia a esta operación), a otra palabra z obtenida al colocar las letras de y a continuación de las de x. Formalmente

$$\text{Sea } x = x_1x_2...x_i,\ y = y_1y_2...y_j \text{ dos palabras }, , x_1, x_2, ..., x_i, y_1, y_2, ..., y_j \in \Sigma$$

$$z = x.y \text{ concatenación de } x \text{ e } y \Leftrightarrow z = x_1x_2...x_iy_1y_2...y_j$$

Lenguajes

Sea Σ un alfabeto. Se llama *lenguaje definido sobre el alfabeto* Σ a cualquier conjunto de palabras formadas con símbolos de Σ.

Sea Σ un alfabeto y L_1 y L_2 dos lenguajes definidos sobre Σ. Se llama *concatenación o producto de* L_1 *y* L_2, y se asignará a esta operación el mismo símbolo que a la concatenación de palabras (el punto ".") al lenguaje formado por todas las concatenaciones de una palabra del lenguaje L_1 y otra del lenguaje L_2. Formalmente

$$L_1 \,.\, L_2 = \{xy | \quad x \in L_1, y \in L_2\}$$

Cuando se concatena un lenguaje consigo mismo se utiliza la notación exponencial con el significado usual de la operación potencia.

La *clausura, cierre o iteración de un lenguaje* L *sobre un alfabeto* Σ, que será representada como L^* se define mediante la siguiente expresión:

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Se utilizará la misma notación para hacer referencia al *conjunto de todas las posibles palabras formadas con los símbolos de un alfabeto* Σ. Este conjunto será representado por la expresión Σ^*.

Reglas de producción y derivaciones

Sea Σ un alfabeto. Se llama *producción* o *regla de producción* a un par ordenado (x, y), donde $x, y \in \Sigma^*$. Se llama *parte izquierda de la regla de producción* a x y *parte derecha de la regla de producción a* y. Es frecuente representar las producciones de la siguiente manera:

$$x ::= y$$

Sea Σ un alfabeto y $x ::= y$ una regla de producción sobre las palabras de ese alfabeto. Sean v y w dos palabras del mismo alfabeto $(v, w \in \Sigma^*)$. Se dice que w *es derivación directa de* v, o que v *produce directamente* w y se expresará

$$v \rightarrow w$$

si existen dos palabras $z, u \in \Sigma^*$, tales que:

$$v = zxu$$

$$w = zyu$$

Sea Σ un alfabeto y P un conjunto de producciones o conjunto de reglas de producción sobre las palabas de ese alfabeto. Sean v y w dos palabras del mismo alfabeto $(v, w \in \Sigma^*)$. Se dice que w *es derivación directa de* v, o que v *produce directamente* w y se expresará

$$v \rightarrow w$$

si existen dos palabras $z, u \in \Sigma^*$, tales que:

$$v = zxu$$

$$w = zyu$$

$$x ::= y \in P$$

Sea Σ un alfabeto y P un conjunto de producciones o conjunto de reglas de producción sobre las palabas de ese alfabeto. Sean v y w dos palabras del mismo alfabeto ($v, w \in \Sigma^*$). Se dice que w *es derivación de* v, o que v *produce* w y se expresará

$$v \rightarrow +w$$

si existe una secuencia finita de palabras cuyo primer elemento sea v y cuya último elemento sea w mediante las cuales se puede encadenar una serie de derivaciones directas. Formalmente

$$v \rightarrow +w \quad \Leftrightarrow \quad \exists\,(x_0, x_1, ..., x_n)\,,\, n > 0 \quad | \; v = x_0 \wedge x_n = w \wedge u_i \rightarrow u_{i+1} \quad \forall i \in [0, n-1] \cap \aleph$$

A la secuencia anterior se la llama *derivación de longitud* n.

Para incluir la posibilidad de que $v = w$ se utilizará la siguiente notación

$$v \rightarrow *w \Leftrightarrow v \rightarrow +w \vee v = w$$

Gramáticas formales y sus lenguajes asociados

Intuitivamente las gramáticas formales contienen toda la información necesaria para construir lenguajes sobre alfabetos de símbolos.

Se llama *gramática formal* G a la cuádrupla

$$G = (\Sigma_T, \Sigma_N, s, P)\,,\, \Sigma_T \cap \Sigma_N = \Phi \wedge s \in \Sigma_N \wedge P = \{u ::= v\,,\, u \in \Sigma^+ \wedge v \in \Sigma^* \wedge u = xAy \wedge x, y \in \Sigma^* \wedge A \in \Sigma_N\}$$

Donde

- Σ_T es el *alfabeto de símbolos terminales,*
- Σ_N es el *alfabeto de símbolos no terminales,*
- s es el *axioma,*
- P es el *conjunto de reglas de producción* .

Y se llamará $\Sigma = \Sigma_T \cup \Sigma_N$.

Sea una gramática $G = (\Sigma_T, \Sigma_N, s, P)$, se llama *lenguaje asociado a* G o *lenguaje generado por* G, y se escribirá $L(G)$, al conjunto

$$L(G) = \{x| \quad s \rightarrow *x \wedge x \in \Sigma_T^*\}$$

La jerarquía de Chomsky

Chomsky clasificó las gramáticas formales en cuatro grupos cada uno de los cuales incluye los anteriores. La figura 2.a muestra un esquema de la jerarquía de Chomsky. La clasificación se realiza en función de las propiedades de las reglas de producción.

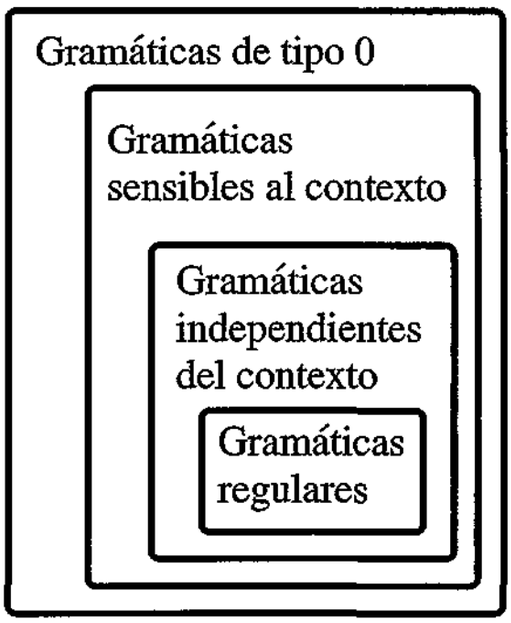


Figura 2.a: Jerarquía de gramáticas de Chomsky.

Las gramáticas de orden 0 son las que no añaden ninguna restricción a la definición general.

Se llama *gramática de orden* 1 a las cuádruplas

$$(\Sigma_T, \Sigma_N, s, P) \,,, \Sigma_T \cap \Sigma_N = \Phi \wedge s \in \Sigma_N \wedge$$

$$P = \{u ::= w\} \,,, u = \begin{cases} s & (1) \\ xAy \,,, x, y \in \Sigma^* \wedge A \in \Sigma_N & (2) \end{cases},$$

$$w = \begin{cases} \lambda & (1) \\ xvy \,,, x, y \in \Sigma^* \wedge v \in \Sigma^+ & (2) \end{cases}$$ Donde (1) (idem. (2)) indica que si la parte izquierda es de tipo (1) (idem. (2)) la parte derecha tiene que ser de tipo (1) (idem. (2)) y viceversa.

Se llama *gramática de orden* 2 a las cuádruplas

$$(\Sigma_T, \Sigma_N, s, P) \,,, \Sigma_T \cap \Sigma_N = \Phi \wedge s \in \Sigma_N \wedge$$

$$P = \{A ::= v\} \,,, A \in \Sigma_N \wedge v \in \Sigma^*$$

Se llama *gramática regular o de orden* 3 a cualquier gramática lineal por la derecha o por la izquierda

- Se llama *gramática de orden* 3 *lineal por la izquierda* a las cuádruplas

$$(\Sigma_T, \Sigma_N, s, P) \,,, \Sigma_T \cap \Sigma_N = \Phi \wedge s \in \Sigma_N \wedge$$

$$P = \{A ::= a \,,, A \in \Sigma_N \wedge a \in \Sigma_T\} \cup \{A ::= Va \,,, A, V \in \Sigma_N \wedge a \in \Sigma_T\} \cup \{s ::= \lambda\}$$

- Se llama *gramática de orden* 3 *lineal por la derecha* a las cuádruplas

$$(\Sigma_T, \Sigma_N, s, P) \;,\; \Sigma_T \cap \Sigma_N = \Phi \wedge s \in \Sigma_N \wedge$$

$$P = \{A ::= a, \;, A \in \Sigma_N \wedge a \in \Sigma_T\} \cup \{A ::= aV, \;, A, V \in \Sigma_N \wedge a \in \Sigma_T\} \cup \{s ::= \lambda\}$$

Los lenguajes también pueden clasificarse por el tipo de la gramática más restrictiva que los puede generar. Tendría sentido, por tanto, hablar de lenguajes de tipo 0, de tipo 1, etc. y estudiar las relaciones de inclusión entre ellos como se ha hecho con las gramáticas formales.

2.1.3 Relación entre máquinas abstractas y lenguajes formales

Las máquinas abstractas y los lenguajes formales resultan ser isomorfos. A cada gramática de la jerarquía de Comsky le corresponde un tipo de máqina que es capaz de reconocerlo y/o generarlo:

- A los lenguajes de tipo 0 le corresponden las máquinas de Turing.
- A los lenguajes de tipo 1 o dependientes del contexto les corresponde los autómatas acotados lineamente.
- A los lenguajes de tipo 2 o independientes del contexto les corresponde los autómatas a pila.
- A los lenguajes de tipo 3 o regulares les corresponde los autómatas finitos.

2.1.4 Relación entre máquinas abstractas y tipos de problemas

También se ha establecido una relación entre los tipos de máquina y los problemas que son capaces de resolver.

Los problemas se presentan clasificados por su complejidad y por el tipo de formalismo necesario para representarlos.

Las expresiones regulares son uno de esos formalismos. Son realmente una notación especial para las operaciones de las palabras construidas con los símbolos de los alfabetos que se está mencionando en estas secciones.

Otro formalismo lo constituyen las funciones recursivas. El concepto de función recursiva fue introducido por Kleene en 1936. Se trata de una clase de funciones matemáticas restringidas a los números naturales (incluído el 0). Al igual que Turing, Kleene pretendía formalizar el concepto de algoritmo. Kleene consiguió una formalización equivalente a la de Turing, es decir, una tarea puede ser realizada por una máquina de Turing si y sólo si puede expresarse como una función recursiva de Kleene.

Sea $\aleph$ el conjunto de los números naturales (incluído el 0). Se llamará *función recursiva* a toda función cuyo origen sea un subconjunto de $\aleph^n$ y cuya imagen sea un subconjunto de $\aleph$, que quede definida mediante los siguientes elementos:

1. Una base de recursión, que establece axiomáticamente ciertos valores de la función para elementos determinados de su conjunto origen.

2. Una regla de construcción recursiva, que permite determinar otros valores de la función a partir de valores conocidos.

3. La afirmación de que la función sólo toma los valores que se obtengan por aplicación, un número finito de veces, de los dos elementos anteriores.

El conjunto de todas las funciones recursivas se puede definir recursivamente mediante la operación de composición de funciones que sean recursivas y la especificación de ciertos elementos básicos que son la función nula, la función sucesor y la función proyección. A todas las funciones mencionadas se les da el significado que habitualmente tienen.

De esta manera, respecto a los tipos de problemas y procediendo en orden de dificultad creciente, puede enunciarse la siguiente lista:

- Los problemas del álgebra de las expresiones regulares pueden ser resueltos por autómatas finitos.
- Los problemas recursivamente enumerables pueden ser resueltos por las máquinas de Turing.
- Los problemas demasiado complicados para ser resueltos por estas máquinas son los no enumerables.

Capítulo 3

Sistemas de Lindenmayer

3.1 Origen de los sistemas L

Los sistemas de Lindenmayer surgieron como un intento de describir de manera matemática el desarrollo de organismos pluricelulares. El objetivo era una formalización en la que se tuviera en cuenta las observaciones puramente morfológicas junto con las aspectos genéticos, citológicos y fisiológicos. Se pretendía proponer sistemas abstractos que sirvieran como marco teórico para estudiar todos estos aspectos.

El modelo (basado en la potencia expresiva de la matemática discreta y la combinatoria) se oponía a la formalización tradicional basada en funciones continuas y ecuaciones diferenciales. El proceso consiste en discretizar tanto el espacio como el tiempo y codificar las reglas de comportamiento de manera simbólica. Ya se conocía los alentadores resultados de disciplinas como la teoría de autómatas y lenguajes formales por lo que se confiaba en la posibilidad de conseguirlo.

Los sistemas de Lindenmayer representan una evolución de otros modelos matemáticos discretos (como los autómatas celulares auto-reproductivos de von Neumann, o los modelos neuronales de McCulloch y Pitts) y, aunque tienen en común bastantes de sus objetos matemáticos difieren en aspectos tan importantes como el tipo de crecimiento permitido a la estructura que representa al organismo.

El enfoque propuesto por Lindenmayer también supone una evolución de la teoría de lenguajes formales de Chomsky como se analizará con más detenimiento tras realizar algunas definiciones más precisas.

3.2 Los distintos tipos de sistemas L

3.2.1 Esquemas 0L

La notación y el modelo básico que Lindenmayer presentó en sus trabajos consideraba las palabras como listas (vectores unidimensionales) de símbolos del alfabeto del sistema. A pesar de ello no hay ninguna limitación teórica para definir sistemas que utilicen palabras en las que los

símbolos se distribuyan en matrices n dimensionales. De hecho el propido Lindenmayer presenta ejemplos de estas características. [*Lin*75].

Intuitivamente los sistemas 0L generalizan el concepto de gramática formal a gramática formal de derivación paralela.

Un objeto matemático que consiste en un conjunto de reglas de producción sin interacciones celulares de un alfaeto de símbolos y de un axioma (la cadena con la que se comienza el proceso) se ha llamado *sistema L sin interacciones o sistema 0L.* Dependiendo de que las reglas de producción asignen una única parte derecha a cada parte izquierda se puede añadir a estos sistemas la característica de ser *sistemas L deterministas sin interacciones o D0L* . Dependiendo de si ninguna regla de producción puede tener en su parte derecha la palabra vacía (λ) se les llamará *sistemas L propagativos* o *sistemas P0L.* Estas dos propiedades pueden producirse simultáneamente por lo que son posibles *sistemas deterministas propagativos de Lindenmayer sin interacciones* o *sistemas PD0L.*

Formalmente,

Se llama esquema 0L a un par

$$S = (\Sigma, P).$$

Donde

- Σ es unconjunto no vacío, *el alfabeto.*
- $P \subseteq \Sigma \times \Sigma^*, S \neq \Phi \mid \ \forall a \in \Sigma \Rightarrow \exists \alpha \in \Sigma^*$,, $(a, \alpha) \in P$, *conjunto de reglas de producción.* Las reglas de producción suelen escribirse indistintamente como (a, α) o como $a ::= \alpha$.

Un esquema 0L $S = (\Sigma, P)$ se llama *propagativo* si no existe en P una producción de la forma $a ::= \lambda$. En otro caso se llama *no propagativo.*

Un esquema 0L $S = (\Sigma, P)$ se llama *determinista* si para todos los símbolos de su alfabeto hay exclusivamente una regla en P que lo tiene como parte izquierda.

3.2.2 Derivación

Descripción intuitiva

La derivación es el concepto que formaliza la manera en la que unas palabras se obtienen a partir de otras por medio de una secuencia de pasos en cada uno de los cuales se utiliza un conjunto de reglas de producción. Es la principal generalización y diferencia con respecto a las gramáticas de la jerarquía de Chomsky. La derivación en las gramáticas de Chomsky es secuencial (en cada paso sólo se modifica una posición de la cadena actual). La derivación en los sistemas de Lindenmayer se realiza en paralelo (en cada paso se modifica simultáneamente todos los símbolos de la cadena actual). Es útil disponer de una notación precisa de esta operación, básica entre cadenas relacionadas por sistemas de Lindenmayer.

Definición

Formalmente.

Sea $S = (\Sigma, P)$ un esquema L , donde los símbolos utilizados ya han sido descritos previamente: Σ es el alfabeto y P el conjunto de reglas de producción.

Una derivación D en S es una terna (O, v, p) donde

- O es un conjunto de pares ordenados de enteros no negativos (las apariciones en D).
- v es una función de O en Σ de forma que $v(i,j)$ es el valor de D en la aparición (i,j).
- p es una función de un subconjunto de O en P de forma que $p(i,j)$ es la regla de producción de D en la aparición (i,j).

que satisface las condiciones siguientes:

- $\exists (x_0, x_1, ..., x_f)$ „ $x_i \in \Sigma^*$ una secuencia de palabras a la que se dará el nombre de *traza de la derivación* y se representará con la expresión $tr(D)$ |
  - (1) $O = \{(i,j) |\ \ 0 \leq i \leq f, \quad 1 \leq j \leq |x_i|\}$
  - (2) $v(i,j) = \pi_j(x_i)$
  - (3) $dom(p) = \{(i,j) |\ \ 0 \leq i \leq f, \quad 1 \leq j \leq |x_i|\}$, donde *dom* representa el dominio de la función.
  - (4) $\forall i,j |\ \ 0 \leq i \leq f, \quad 1 \leq j \leq |x_i| \Rightarrow p(i,j) = v(i,j) ::= \alpha_i$,, $x_{i+1} = \alpha_1\alpha_2...\alpha_{|x_i|}$

En ese caso, se dice que

- D es *una derivación de x_f desde x_0*.
- f es *la profundidad de la derivación* D y se representa como $prof(D)$.
- La cadena x_f Se llama *resultado de la derivación* D y se representa como $\rho(D)$.

Y se escribirá $x_0 \Rightarrow_S x_f$

Ejemplo

Sea $S = (\Sigma = \{a,b\}, P = \{R_1 = a ::= b, \quad R_1 = b ::= ab\})$ un esquema de Lindenmayer.

Los conjuntos O, v y p de la derivación $a \rightarrow b \rightarrow ab \rightarrow bab$ se muestran a continuación:

- $O = \left\{ \begin{array}{c} (0,1), \\ (1,1), \\ (2,1), (2,2) \\ (3,1), (3,2), (3,3) \end{array} \right\}$

- $v = \left\{ \begin{array}{c} ((0,1),a), \\ ((1,1),b), \\ ((2,1),a),\ ((2,2),b) \\ ((3,1),b),\ ((3,2),a),\ ((3,3),b) \end{array} \right\}$
- $p = \left\{ \begin{array}{c} ((0,1),R_1), \\ ((1,1),R_2), \\ ((2,1),R_1),\ ((2,2),R_2) \end{array} \right\}$

Además:

- $tr\,(D) = (a,\ b,\ ab,\ bab)$
- $\rho\,(D) = bab$
- $prof\,(D) = 3$

3.2.3 Lenguaje definido por un esquema 0L

Al igual que después de las gramáticas de Chomsky se definían los lenguajes que tienen asociados; tras definir los esquemas L hay que definir sus lenguajes asociados.

Sea $S = (\Sigma, P)$ un esquema 0L.

Sea $x \in \Sigma^*$ palabra construída con los símbolos de Σ.

Sea $n \in \aleph$.

Se define por inducción el *lenguaje definido por el esquema S y la palabra x en n pasos:*

$$L_0\,(S,x) = \{x\}$$

$$L_{n+1}\,(S,x) = \{y|\ \ \exists z,, z \in L_n\,(S,x) \wedge z \Rightarrow_S y\}$$

El *lenguaje definido por el esquema S y la palabra x* es la unión de de los lenguajes definidos por S y x en cualquier número de pasos. Formalmente:

$$L\,(S,x) = \bigcup_{i=0}^{\infty} L_i\,(S,x)$$

3.2.4 Sistemas 0L

Un *sistema 0L* es una terna

$$G = (\Sigma, P, w)$$

Donde $S = (\Sigma, P)$ es un esquema 0L y w, *el axioma del sistema,* es una palabra construída con símbolos de Σ. Se dice que es *un sistema determinista* si su esquema lo es y se dice que es *propagativo* si su esquema lo es. Cuando se utilice con sistemas L conceptos y notaciones definidos en esquemas L, se supondrá que se están aplicando al esquema del sistema L. Esta obsesrvación es extensiva a cualquier tipo de sistema L.

3.2.5 Sistemas $\langle k,l\rangle$IL

Los sistemas L con interacciones extienden el modelo de derivación en paralelo a las gramáticas de Chomsky dependientes del contexto. Es decir, se especifica reglas de producción en cuya parte izquierda hay cadenas en lugar de símbolos. Esas cadenas están formadas por el símbolo que la regla va a transformar y el contexto en el que la transformación es aplicable. El contexto puede ser por la derecha o por la izquierda. Todas las reglas tienen que considerar el mismo número de símbolos como contexto. Esto puede plantear problemas en los extremos, al intentar aplicar reglas de producción a los símbolos que estén demasiado cerca del principio o el final de la cadena. Se utiliza un símbolo totalmente nuevo que sólo se usa para rellenar correctamente el contexto en estos casos y poder especificar reglas que tengan el mismo tamaño de contexto que las demás. Dicho símbolo sólo puede aparecer en los extremos.

Los sistemas con estas características se llaman *sistemas de Lindenmayer con interacciones* (k,l) donde k y l son las longitudes de las que se hablaba en la frase anterior.

Formalmente:

Sean k y l dos enteros no negativos. *Un sistema* $\langle k,l\rangle$ *de Lindenmayer con interacciones o sistema* $\langle k,l\rangle IL$ es una cuadrupla $G=(\Sigma,P,g,w)$.

Donde,

- Σ es un conjunto finito y no vacío (el *alfabeto de* G).
- $w\in\Sigma^*$ (el *axioma de* G).
- $g\notin\Sigma$ (es el *elemento de marca de* G, que sirve para completar el contexto en los extremos de las palabras).
- $P\subset\left((\Sigma\cup\{g\})^k\times\Sigma\times(\Sigma\cup\{g\})^l\right)\times\Sigma^*$ es una relación no vacía y finita entre los conjuntos de cadenas de símbolos compuestos por k símbolos de contexto a la izquierda, l la derecha y un símbolo del alfabeto y las cadenas de símbolos del alfabeto; que cumple las dos condiciones (1) y (2) siguientes (obsérvese que la condición (1) tiene como consecuente la conjunción de dos condiciones (1.1) y (1.2) relacionadas con dos cadenas w_1,w_3 y un símbolo del alfabeto a, suele decirse de las ternas (w_1,a,w_3) que cumplen estas dos condiciones que son *aplicables*):
  - (1) Si $(w_1,a,w_3,w_4)\in P$, entonces
    - (1.1) Si $\exists\overline{w_1},\overline{\overline{w_1}}\in(\Sigma\cup\{g\})^*\mid\ w_1=\overline{w_1}g\overline{\overline{w_1}}\Rightarrow\overline{w_1}\in\{g\}^*\wedge$
    - (1.2) Si $\exists\overline{w_3},\overline{\overline{w_3}}\in(\Sigma\cup\{g\})^*\mid\ w_3=\overline{w_3}g\overline{\overline{w_3}}\Rightarrow\overline{\overline{w_3}}\in\{g\}^*$
    - Las ternas (w_1,a,w_3) que cumplen (1.1) y (1.2) se llaman *aplicables*.
  - (2) $\forall(w_1,a,w_3)\in\left((\Sigma\cup\{g\})^k\times\Sigma\times(\Sigma\cup\{g\})^l\right)\mid(w_1,a,w_3)$ es aplicable (satisface (1.1) y (1.2)) $\exists w_4\in\Sigma^*,,(w_1,a,w_3,w_4)\in P$.
- Obsérvese que:
  - La condición (1) asegura que hay reglas para la transformación de los símbolos a una distancia menor de k caracteres del principio de la cadena o l del final. Además el

símbolo "g" no puede estar en medio de la cadena, todas sus apariciones tienen que estar agrupadas al principio o al final de la misma.

- La condición (2) asegura que hay reglas de producción para todos los contextos posibles.

Cuando sea más cómodo referirse a las componentes de un sistema IL G en función del propio sistema se utilizará la siguiente notación.

- $alfabeto\,(G)$
- $axioma\,(G)$
- $reglas\,(G)$
- $esquema\,(G) = (alfabeto\,(G)\,, reglas\,(G))$
- $marcador\,(G)$

3.2.6 Sistemas con tablas

También es interesante para simular algunos procesos, considerar más de un conjunto de reglas de producción que se pueden aplicar en circunstancias distintas. Estos sistemas se llaman *sistemas de Lindenmayer con tablas* o *sistemas TL.*

Se llama *esquema T0L* a un par

$$S = \left(\Sigma, \widehat{P}\right)$$

Donde

- Σ es el *alfabeto del sistema.*
- $\widehat{P}$ es el *conjunto de tablas de* S, un conjunto no vacío de tablas de producción de forma que cada uno de sus elementos P es un conjunto de reglas de producción sobre el alfabeto Σ.

Todos los conceptos y notaciones definidos para sistemas y esquemas de Lindenmayer son aplicables a los sistemas y esquemas implícitos que el conjunto $\widehat{P}$ define, ya que cada elemento suyo junto con el alfabeto Σ define un esquema 0L.

3.2.7 Sistemas con extensiones

Para simular algunos sistemas biológicos resulta conveniente especificar un subconjunto de los alfabetos de los sistemas L para que sólo las palabras formadas con símbolos de ese subalfabeto sean consideradas como el lenguaje generado por el sistema L. En estos casos se considera que los sistemas tienen *extensiones.* Se llamará *sistema E0L* a un *sistema sin interacciones con extensiones.* Se llamará *sistema EIL* a un *sistema de Lindenmayer con interacciones y con*

extensiones. Se llamará *sistema ET0L* a un *sistema de Lindenmayer sin interacciones, con tablas y con extensiones.*

Formalmente:

Un *sistema E0L* es una cuádrupla

$$G = (\Sigma, P, w, \Delta)$$

Donde

- $\overline{G} = (\Sigma, P, w)$ es un sistema 0L
- $\Delta \subset \Sigma$.
- $L(G) = L(\overline{G}) \cap \Delta^*$

3.2.8 Otras combinaciones

Las definiciones que se han presentado son las de las familias básicas de cada tipo. Debe utilizarse la misma notación para describir otros sistemas cuyas propiedades no coincidan con las desarrolladas completamente en estas páginas.

3.3 Diferencias entre los sistemas L y las gramáticas de Chomsky

Los sistemas de Lindenmayer y las gramáticas de Chomsky se diferencian esencialmente en el objetivo de las mismas. Las gramáticas de Chomsky pretenden formalizar la estructura subyacente en los lenguajes. Es esencial en estas gramáticas determinar qué palabras representan las oraciones del lenguaje y distinguirlas claramente de los pasos intermedios. Estos pasos son necesarios para la construcción estructurada de oraciones correctas; por su característica de paso intermedio, no tienen que tener necesariamente sentido como oraciones del lenguaje. Los sistemas de Lindenmayer, sin embargo, pretenden describir el proceso de crecimiento de los organismos vivos. Por lo tanto todos los pasos son igualmente importantes.

Esto hace que se puedan apreciar, como habrá ocurrido tras la detallada comparación de las definiciones formales, las siguientes diferencias:

Primero, el objetivo de los sistemas de Lindenmayer de simular el desarrollo de seres vivos hace que su concepto de derivación difiera sensiblemente del de Chomsky. En las gramáticas de Chomsky, la derivación directa implica la selección de un único lugar de la cadena y una única regla de producción para ser aplicada cada vez. En las derivaciones de los sistemas de Lindenmayer se aplican reglas de producción a todos los símbolos de una cadena para obtener la siguiente. Así se simula con más fidelidad el proceso de crecimiento que se realiza simultáneamente en todas las partes del organismo.

El interés en el proceso en lugar de en su resultado hace que en los sistemas de Lindenmayer ninguno de los símbolos sea designado como no terminal. El concepto de lenguaje generado por

un sistema L es, por lo tanto, menos limitado porque incluye en general a todas las cadenas que puedan derivarse a partir de su axioma.

La figura 2.b muestra la jerarquía existente entre los sistemas definidos según la relación de inclusión. Dos familias de sistemas están unidas por un segmento ascendente cuando la inferior está incluida en la superior.

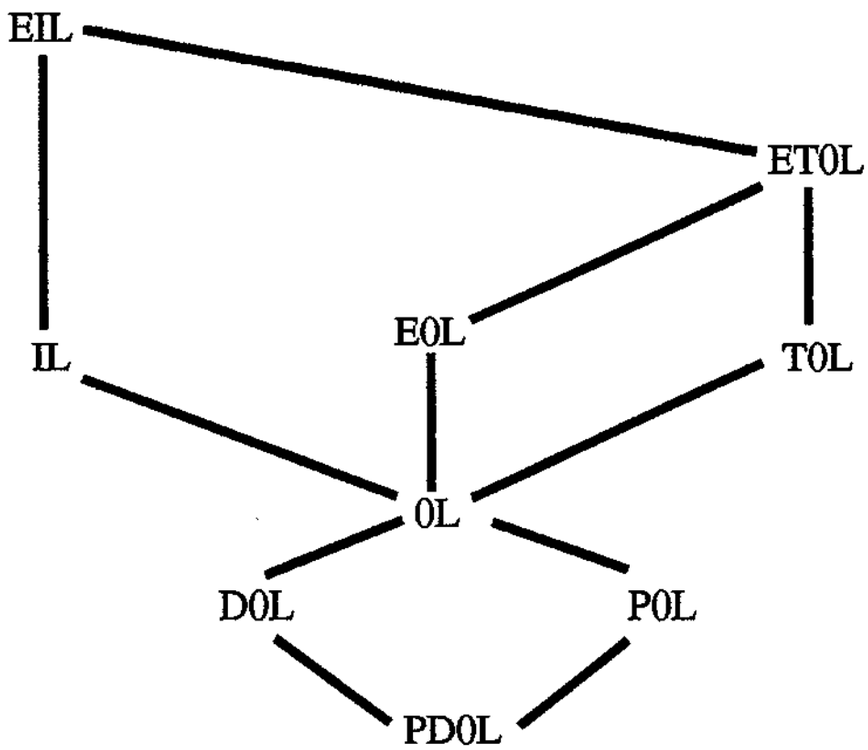


Figura 2.b: Relaciones de inclusión entre sistemas L.

3.4 Comparación entre sistemas L y gramáticas de Chomsky

Una vez presentados los tipos de lenguajes de Chomsky y los de Lindenmayer y las relaciones de inclusión que hay en ambos grupos, puede resultar interesante mencionar brevemente las relaciones de inclusión que relacionan todos con todos.

La figura 2.c muestra un esquema de la jerarquía de inclusión.

Se utilizará las siguientes abreviaturas:

- RE, de *r*ecursivamente *e*numerables, para los lenguajes de gramáticas de tipo 0.
- DC, de *d*ependientes del *c*ontexto, para los lenguajes de gramáticas de tipo 1.

- IC, de *i*ndependientes del *c*ontexto, para los lenguajes de gramáticas de tipo 2.
- RG, de *r*e*g*ulares, para los lenguajes de gramáticas de tipo 3.

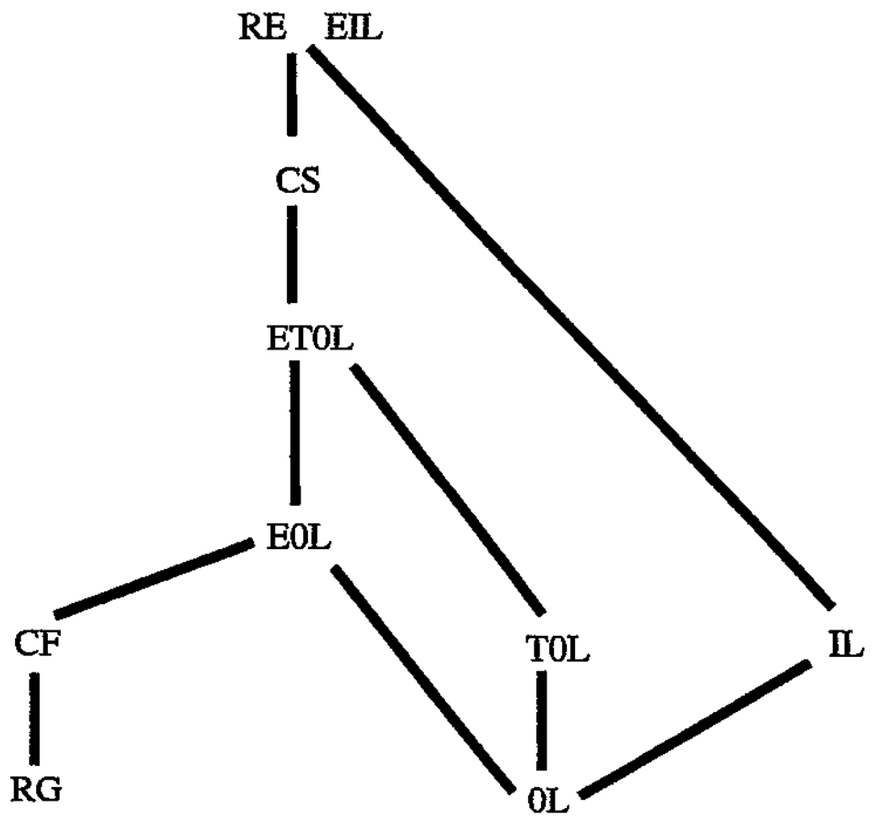


Figura 2.c: Relaciones de inclusión entre sistemas L y de Chomsky.

3.5 Algunas líneas actuales en el uso de sistemas L

Áreas en las que el esfuerzo de investigación y desarrollo se ha incrementado sensiblemente en los últimos años son la simulación y la vida artificial. Nuevos enfoques han sido posibles mediente el uso (y a veces el redescubrimiento) de modelos matemáticos discretos y simbólicos basados en lenguajes formales y máquinas abstractas como las presentadas en estas páginas. Uno de los campos de aplicación de los sistemas L, más explorados desde su aparición fue la generación de imágenes realistas. En la actualidad se está utilizando estos sistemas (y variaciones suyas) para obtener no sólo gráficas que simulen el aspecto de objetos reales sino réplicas (modelos teóricos) que simulen con suficiente fidelidad los procesos reales.

3.6 Aspectos de los sistemas L tratados en la tesis

Nuestro trabajo tiene como objetivo analizar el poder expresivo de los sistemas de Lindenmayer intentando establecer correspondencias con otros formalismos utilizados para representar sistemas complejos.

Capítulo 4

Autómatas Celulares

4.1 Origen de los autómatas celulares

Los autómatas celulares fueron definidos por von Neumann y Ulam en la década de los 60 como una posible formalización de la propiedad de ciertos sistemas biológicos de auto reproducirse aunque, posteriormente, han sido de utilidad en muchas otras tareas.

Los autómatas celulares son una abstracción matemática de sistemas físicos en los que el espacio, el tiempo y las variables que describen el estado del sistema son discretos. Un autómata celular consta de un retículo uniforme regular habitualmente de extensión infinita (rejilla), en cada una de cuyas celdas hay un autómata finito, es decir, una variable discreta que cambia de estado en función de la entrada que recibe del exterior. El estado del autómata celular queda totalmente descrito mediante el valor de los autómatas finitos de cada celda. El autómata celular evoluciona a intervalos discretos de tiempo, el valor de cada autómata finito en un instante determinado depende de los valores de sus vecinos en el instante anterior. El vecindario de una posición habitualmente está formado por el autómata de la posición considerada y los autómatas adyacentes. El estado de todas las posiciones es actualizado simultáneamente de forma que el comportamiento del autómata celular está definido exclusivamente en función de reglas de transición locales.

Las rejillas pueden ser de diferentes dimensiones y formas aunque las más estudiadas son las cadenas unidimensionales y las rejillas rectangulares bidimensionales. Algunos conjuntos de autómatas celulares han sido analizados de forma exhaustiva (por ejemplo, los autómatas celulares unidimensionales binarios con vecindario formado por el autómata estudiado, el que le antecede y el que le precede en la cadena [*Wol*94]) y se ha podido comprobar la complejidad de su comportamiento. A partir de distribuciones aleatorias de ceros y unos se llega a configuraciones estables ya sea constantes o con un período más o menos largo. Se ha constatado también su potencia expresiva. Algunos autómatas bidimensionales son computacionalmente completos (tanto undimensionales [*Smt*71] como bidimensionales, por ejemplo el juego de la vida de Conway [*Cnw*]). Los autómatas celulares son sistemas dinámicos complejos, se han utilizado como alternativa a la construcción de modelos físicos basados en sus ecuaciones diferenciales.

En las próximas secciones se especifica la notación usada en la presente tesis.

4.2 Descripción informal de rejilla

En los autómatas celulares se distribuye un conjunto de autómatas finitos sobre una estructura de topología regular generalmente asociada a una especie de matriz sin limitación en el número de dimensiones y con la posibilidad de no ser finita.

4.3 Definición

Dado un conjunto E.

Dados n conjuntos de índices no necesariamente finitos (subconjunto creciente de números enteros contiguos cuyo mínimo es necesariamente 0) $\{I_i\}_{i=0}^{n-1}$,, $I_i \subseteq Z \quad \forall i$.

Una *rejilla n-dimensional sobre E* es cualquier función

$R : I_0 \times I_1 \times ... \times I_{n-1} \to E$,, $R(i_0, ..., i_{n-1}) = a_{i_0,...,i_{n-1}} \circ a[i_0, ..., i_{n-1}]$ que serán las dos notaciones utilizadas $\forall (i_0, ..., i_{n-1}) \in I_0 \times ... \times I_{n-1}$.

4.4 Notación

Con las definiciones anteriores, se escribirá

$$R_{a_0,...,a_{n-1}} \text{ sobre } E \text{ donde } a_i = \begin{cases} 1 + \max(I_i) & si \quad I_i \quad finito \\ \infty & en \quad otro \quad caso \end{cases}$$

para referirse al conjunto de todas las rejillas de esas dimensiones.

4.5 Ejemplos

- R_∞ sobre $\Re$, podría utilizarse para representar todos los vectores de infinitos números reales.
- $R_{\infty,\infty}$ sobre $\{0, 1\}$ representa rejillas bidimensionales infinitas booleanas.
- $R_{3,\infty}$ sobre $\{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ representa rejillas bidimensionales de tres filas infinitas (en su número de columnas) octales.
- $R_{3,3,3}$ sobre Z representa rejillas tridimensionales de números enteros y de dimensión $3 \times 3 \times 3$.

4.6 Descripción informal

Es frecuente, en la definición de los autómatas celulares, que se simule rejillas infinitas mediante rejillas finitas. Para ello se define lo que se conoce como *condiciones de contorno*. Se asigna a cada posición externa a la rejilla una de su interior.

4.7 Definición

Supóngase las definiciones anterioes.

Sea $I_F = \{I_j^F\}_{j\in\{0,\ldots,p-1\}}$ la familia de los p índices finitos de una rejilla n-dimensional cualquiera.

Una *condición de contorno* c es cualquier función

$$c : \widehat{I_0} \times \ldots \times \widehat{I_{n-1}} \rightarrow I_0 \times \ldots \times I_{n-1} \;,\; \widehat{I_i} = \begin{cases} I_i & si \quad I_i \in I_F \\ Z & otro \quad caso \end{cases}$$

4.8 Definición

La condición de contorno de toro plano o de rosquilla para una rejilla bidimensional ($R \in M_{m,n}$) es la siguiente función

$$c_{T2} : Z \times Z \rightarrow [0, m-1] \times [0, n-1] \;,\; c_{T2}((i,j)) = (i \mod m, j \mod n)$$

4.9 Ejemplo

La figura 7.0.a muestra la condición de contorno de toro plano (c_{T2}) para una rejilla de $M_{2,2}$ y para las filas y columnas exteriores más cercanas a la matriz. La rejilla se muestra resaltada en el centro.

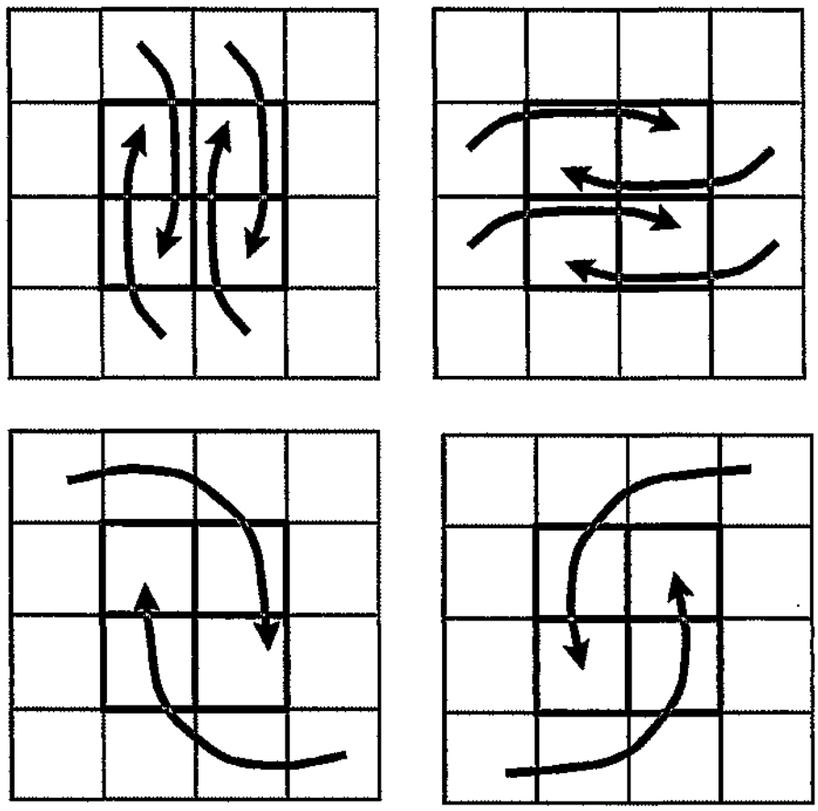


Figura 7.0.a Condición de contorno c_{T2}: posiciones más próximas.

4.10 Notación de autómata finito determinista

Los autómatas finitos deterministas fueron descritos en la sección 2.1.1. Cuando se quiera hacer mención a las componentes de un autómata se utilizará la siguiente notación:

Si A es el autómata

- *entrada* (A) es el alfabeto de entrada.
- *estados* (A) es el conjunto de estados.
- *transicion* (A) es la función de transición de estados.
- *inicial* (A) es el estado inicial.
- *objetivo* (A) es el conjunto de estados objetivo.
- *estado* (A) es el estado actual.

4.11 Definición

Sea R una rejilla n-dimensional de autómatas.

Sea Q el conjunto de todos los posibles estados de los autómatas de R.

Se llama *configuración de R y Q en un instante t*, y se escribe $C(R,Q,t)$, o simplemente $C(t)$, si no hay ambigüedad respecto a la rejilla y el conjunto de estados considerados, a una función inyectiva C dependiente del tiempo que asigna en cada instante un estado a cada autómata de la rejilla.

Formalmente

$$C(R,Q,t) : R \to Q$$

Se llama *configuración actual de una rejilla de autómatas* R y se escribe $C_a(R)$ a la que asigna a cada autómata de R su estado actual.

4.12 Definición

Una *vecindad n-dimensional* V es un par (k,N) que se utiliza para determinar los vecinos de cada posición de cualquier rejilla n-dimensional M.

Formalmente,

- $V = (k,N)$, donde
  - $k \in \aleph$, es el número de vecinos de cualquier posición de una rejilla.
  - N, es un vector de n desplazamientos que, sumados a cualquier posición de la rejilla, determina el conjunto de sus posiciones vecinas. Es decir, cada una de las k componentes de N (una por cada vecino) es un vector de n deplazamientos (uno por cada dimensión de la rejilla) que sumado al vector utilizado como índice para localizar una posición de la rejilla produce un vector que puede ser utilizado como índice para localizar cada vecino. Formalmente,

$$N \in Z^{n^k}$$

Si $\overrightarrow{i} = (i_0, ..., i_{n-1})$ es el índice de una posición de la rejilla.

$$\text{Si } N = \begin{pmatrix} \overrightarrow{\delta_0} = (\delta_0^0, ..., \delta_{n-1}^0) \\ \vdots \\ \overrightarrow{\delta_k} = (\delta_0^k, ..., \delta_{n-1}^k) \end{pmatrix}$$

El vector de posiciones de los vecinos se puede calcular mediante la siguiente suma de la siguiente manera:

$$\begin{pmatrix} \overrightarrow{\delta_0} + \overrightarrow{i} \\ \vdots \\ \overrightarrow{\delta_k} + \overrightarrow{i} \end{pmatrix} = \begin{pmatrix} (i_0 + \delta_0^0, ..., i_{n-1} + \delta_{n-1}^0) \\ \vdots \\ (i_0 + \delta_0^k, ..., i_{n-1} + \delta_{n-1}^k) \end{pmatrix}$$

El vecindario de la posición $\vec{i}$, que se escribirá *vecindario* $\left(V, \vec{i}\right)$ o, simplemente, *vecindario* $\left(\vec{i}\right)$ si no hay ambigüedad respecto a la vecindad considerada, es el conjunto de vecinos de la posición estudiada. Donde $\vec{i}$ es un índice válido que determina una posición en la rejilla G.

Se utilizará una representación vectorial o en forma de conjunto de *vecindario* $\left(\vec{i}\right)$, dependiendo de si conviene resaltar el orden de los elementos o no.

Formalmente,

$$\vec{i} \in Z^n$$

$$\textit{vecindario}\left(\vec{i}\right) = \left(G\left[\vec{i} + \vec{\delta_0}\right], ..., G\left[\vec{i} + \vec{\delta_0}\right]\right) \quad \forall \left(\vec{\delta_i}\right) \in N \text{ (como vector)}$$

$$\textit{vecindario}\left(\vec{i}\right) = \left\{G\left[\vec{i} + \vec{\delta_i}\right] \quad \forall \left(\vec{\delta_i}\right) \in N\right\} \text{ (como conjunto)}$$

4.13 Observación

Sea R una rejilla n-dimensional de autómatas.

Sea Q el conjunto de todos los estados posibles de los autómatas de R.

Sea V una vecindad n-dimensional que define un *vecindario* $\left(\vec{i}\right)$ $\forall \vec{i}$ índice válido sobre R.

Sea C una configuración definida sobre R.

C define para cualquier índice $\vec{i}$ sobre R *la configuración de su vecindario* mediante la restricción de C a *vecindario* $\left(\vec{i}\right)$.

4.14 Definición

La vecindad bidimensional de von Neumann, que se escribe V_N, es la vecindad bidimensional que elige la posición estudiada y los cuatro vecinos más próximos a ella según la distancia euclídea.

Formalmente,

- $V_N = (5, ((0,0), (0,1), (1,0), (0,-1), (-1,0)))$

La figura 7.0.b muestra la disposición de esta vecindad en la rejilla :

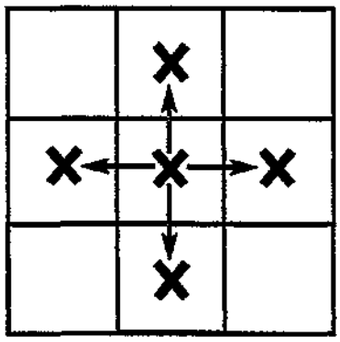


Figura 7.0.b Vecindad bidimensional de von Neumann de la posición central.

4.15 Definición

La vecindad tridimensional de von Neumman, que se escribe V_{N^3}, es la vecindad tridimensional que elige la posición estudiada y los seis vecinos más próximos a ella según la distancia euclídea.

Formalmente,

- $V_N = (7, ((0,0,0), (0,1,0), (1,0,0), (0,-1,0), (-1,0,0), (0,0,1), (0,0,-1)))$

La figura 7.0.b.2 muestra la disposición de esta vecindad en la rejilla :

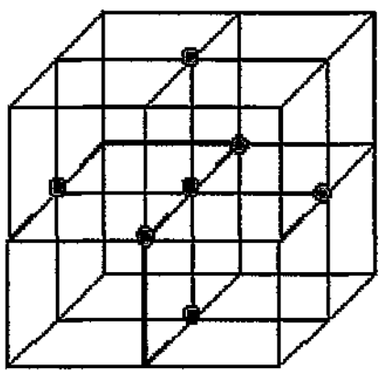


Figura 7.0.b.2 Vecindad tridimensional de von Neumann de la posición central.

4.16 Definición

La vecindad bidimensional de Moore, que se escribe V_M, es la vecindad bidimensional que elige la posición estudiada y los ocho vecinos más próximos a ella según la distancia euclídea.

Formalmente,

- $V_M = (9, ((0,0), (-1,1), (0,1), (1,1), (1,0), (1,-1), (0,-1), (-1,-1), (-1,0)))$

La gráfica 7.0.c muestra la disposición de esta vecindad en la rejilla:

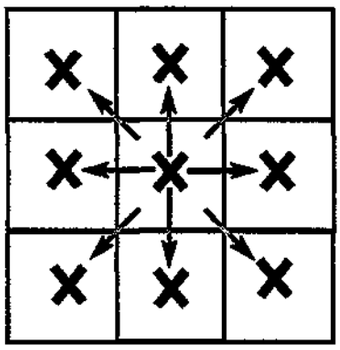


Figura 7.0.c Vecindad bidimensional de Moore de la posición central.

4.17 Definición

Un *autómata celular n-dimensional determinista* es la séxtupla

$$(G, G_0, V, Q, f, T),$$

Donde,

- G es una rejilla n-dimensional de autómatas.
- Q es el *conjunto de estados*: un conjunto finito y no vacío de posibles estados de cada autómata de la rejilla.
- G_0 representa *el estado inicial del autómata celular n-dimensional determinista* y es la configuración inicial ($t = 0$) de G y Q. Asigna un estado inicial a cada autómata de la rejilla. Formalmente.

$$G_0 : G \to Q$$

- V (la *vecindad*) es una vecindad n-dimensional que determina el conjunto de los vecinos de cada autómata de G.
- f es la *función de transición*, es una función que asigna a cada autómata de la rejilla junto a sus vecinos el nuevo estado del autómata estudiado. Formalmente:
  - $f : Q \times Q^n \to Q$

* Donde $f\left(C_a\left(G\left[\vec{i}\right]\right), C_a\left(vecindario\left(\vec{i}\right)\right)\right)$ es el próximo estado que le corresponde al autómata de la posición $\vec{i}$, dados su configuración actual y la de su vecindario.

- $T \subseteq Q$ es el conjunto de estados finales.

Todos los autómatas de la rejilla tienen la misma manera de elegir sus vecinos, la misma función de transición y el mismo conjunto de estados finales.

4.18 Ejemplo

4.18.1 El juego de la vida de Conway

El juego de la vida es un modelo muy simple de comportamiento biológico en un territorio bidimensional dividido en cuadrículas. Cada cuadrícula puede contener un individuo o ninguno. El nacimiento y muerte de los individuos sigue las siguientes reglas: el nacimiento se produce sólo si hay tres sujetos más alrededor, la muerte puede ser debida al aislamiento (hay menos de 2 alrededor) o a la superpoblación (hay más de 3).

Parte del interés y popularidad de este juego son debidos a que se ha demostrado que, a pesar de su simplicidad, el juego de la vida es computacionalmente completo.

Formalmente:

El siguiente autómata celular:

$$A_V = \begin{pmatrix} R_V \in R_{\infty,\infty}, \\ G_{0V}, \\ V_M, \\ Q_V = \{0,1\}, \\ f_V, \\ T = \Phi \end{pmatrix}$$

Donde

- R_V es una rejilla bidimensional infinita, que habitualmente es representada mediante una rejilla bidimensional finita con condiciones de contorno de toro plano (c_{T2}). En la figura 7.0.d se muestra un ejemplo generado mediante ordenador en el que se ha utilizado una $R'_V \in M_{28,33}$ con la condición de contorno c_{T2}.
- Se utiliza la vecindad de Moore.
- G_{0V} asigna arbitrariamente un elemento de Q_V a cada posición de la rejilla R_V (o R'_V) .
- f_V cambia el estado de los autómatas con estado 0 a 1 si 3 de sus vecinos están a 1 y mantiene el estado de los autómatas con estado 1 si 2 o 3 de sus vecinos también están a 1. Formalmente:

$$f_V\left(0,\left(q_{00}=0, q_{-11}, q_{01}, q_{11}, q_{10}, q_{1-1}, q_{0-1}, q_{-1-1}, q_{-10}\right)\right) = \begin{cases} 1 & si \quad \#\{q_{ij}| \quad i \neq 0 \wedge j \neq 0 \wedge q_{ij} = 1\} = 3 \\ 0 & otro \quad caso \end{cases}$$

$$f_V(1,(q_{00}=1,q_{-11},q_{01},q_{11},q_{10},q_{1-1},q_{0-1},q_{-1-1},q_{-10})) =$$
$$\begin{cases} 1 & si \quad \#\{q_{ij} |\ i\neq 0 \wedge j\neq 0 \wedge q_{ij}=1\} \in \{2,3\} \\ 0 & otro \quad caso \end{cases}$$

- T está vacío.

En la figura 7.0.d se muestra la rejilla con la configuración inicial en $t = 0$ (izquierda; las cuadrículas vacías representan autómatas finitos con estado 0, las llenas estado 1) y la configuración siguiente en $t = 1$.

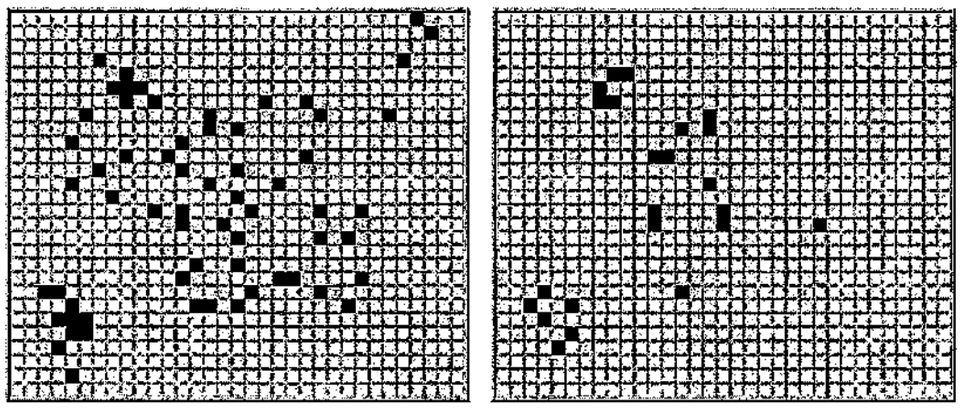


Figura 7.0.d Ejemplo juego de la vida de Conway: dos primeras configuraciones.

4.19 Notación

Dado un autómata celular n-dimensional determinista A se utilizará la siguiente notación para referirse a sus componentes. Obsérvese que sólo se añade nueva notación para aquello que no se pueda utilizar a partir de la notación de autómatas finitos deterministas.

- *rejilla* (A) para referirse a la componente que en la definición se llama G.
- *dimension* (A) es la dimensión de *rejilla* (A)
- *poblacion* (A) es el número de vecinos, primera componente de la vecindad de A.
- *desplazamientos* (A) es el vector de desplazamientos, segunda componente de la vecindad de A.
- *vecindad* (A) es la vecindad utilizada por el autómata celular, es decir, se cumple que *vecindad* (A) = (*poblacion* (A) , *desplazamientos* (A)).

4.20 Definición

Un *autómata celular n-dimensional no determinista* es la séxtupla

$$(G, G_0, V, Q, f, T),$$

Donde,

- G, Q, G_0, V y T han sido defindos previamente.
- f es la *función de transición*, es una función que asigna a cada autómata de la rejilla junto a sus vecinos el conjunto de los nuevos estados posibles para el autómata estudiado. Formalmente:
  - $f : Q \times Q^n \to \wp(Q)$
    - Donde $f\left(C_a\left(G\left[\vec{i}\right]\right), C_a\left(vecindario\left(\vec{i}\right)\right)\right)$ es el conjunto de los posibles estados a los que el autómata de la posición $\vec{i}$ puede transitar, dados su configuración actual y la de su vecindario.
  - $T \subseteq Q$ es el conjunto de estados finales.

4.21 Ejemplo

En el capítulo 9 puede encontrarse algún ejemplo de autómata n-dimensional no determinista.

4.22 Notación

Dado un autómata celular $A = (G, G_0, V, Q, f, T)$, resulta evidente que cada autómata a de la rejilla G se deduce de las definiciones anteriores de la siguiente forma:

$$a = (Q^{poblacion(A)}, Q, f, G_0(a), T)$$

Es decir,

- El alfabeto de entrada es el conjunto finito no vacío $Q^{poblacion(A)}$,recuérdese que $poblacion\,(A)$ es el número de vecinos de cada autómata de A. Por lo tanto el alfabeto es un conjunto de tuplas de estados. Cada tupla es una posible asignación de estados a los vecinos de el autómata considerado. Realmente esta asignación de estados es la entrada que recibe cada autómata.
- El conjunto de estados posibles es el mismo que para el autómata celular: Q.
- El estado inicial es el que devuelve la función del autómata celular: $G_0\,(a)$.
- La función de transición f es la misma del autómata celular. Ahora las tuplas de símbolos de Q son los símbolos de entrada del autómata finito.
- El conjunto de estados objetivo es el mismo que para el autómata celular: T.

4.23 Descripción intuitiva de los autómatas celulares probabilistas

En todo autómata celular determinista A_D, dada la configuración actual de cada estado de la rejilla y de su vecindario, existe un único estado al que cada autómata puede transitar determinado por $transicion\,(A_D)$.

En todo autómata celular no determinista A_N, $transicion\,(A_N)$ sólo especifica un conjunto de opciones de transición pero no aporta información respecto a la manera en la que se realiza la elección en cada instante.

Los autómatas celulares probabilistas añaden a cada una de las opciones de transición la probabilidad con la que son elegidas.

Para ello es necesario modificar la definición del autómata celular A en los siguientes aspectos:

- La manera en la que se asocia estados iniciales a la rejilla ($inicial\,(A)$). En los autómatas estudiados hasta el momento se utilizaba una configuración inicial (G_0) que asignaba de forma explícita un estado concreto a cada autómata de $rejilla\,(A)$. En los autómatas celulares probabilistas G_0 debe asignar a cada posición de la rejilla un vector de probabilidades con una componente por cada posible estado (elemento de Q) que indique la probabilidad de que el autómata que está en esa posición tome inicialmente como estado el elemento de Q asociado a la posición considerada.
- La función de transición f de los autómatas anteriores es reemplazada por un conjunto de matrices estocásticas que asignan probabilidad a cada una de las posibles transiciones.

4.24 Definición

Sea X una variable aleatoria discreta con valores finitos en $X = \{x_0, ..., x_{\#X-1}\}$.

Se llamará a $p(x_i) = p(X = x_i)$ función de probabilidad de la variable aleatoria X, si satisface las siguientes propiedades:

$$p(x_i) \geq 0 \quad \forall x_i \in X$$
$$\sum_{i=0}^{\#X-1} p(x_i) = 1$$

Un *vector de estado* es una representación vectorial de estas funciones de probabilidad.

La descripción formal de un vector de estado (v_X) para la variable aleatoria discreta finita X es la siguiente

$$v_X \in ([0,1] \bigcap \Re)^{\#X} \,,\, \sum_{i=0}^{\#X-1} \prod_i (v_X) = 1,$$

es decir,

$$v_X = (p(x_0), ..., p(x_i), ..., p(x_{\#X-1}))$$

Donde,

- $\prod_i (v_X)$, representa la proyección i-ésima del vector v_X .

A lo largo de este trabajo se supondrá que los vectores estados están ordenados según la enumeración de los valores de la variable aleatoria. Por tanto, se utilizará indistantemente las siguientes notaciones.

$$\prod_i (v_X) = \prod_{x_i} (v_X) = p(x_i)$$

4.25 Definición

Un *autómata celular n-dimensional probabilista* A es la séxtupla

$$(G, G_0, V, Q, F, T),$$

Donde,

- G, Q, V y T han sido defindos previamente.
- G_0 es una función inyectiva que asigna un vector de estado (el vector de estado inicial $G_0(x)$) para Q a cada autómata x de la rejilla G. Formalmente, $G_0 : G \to ([0,1] \bigcap \Re)^{\#Q}$. Es decir, $\prod_i (G_0(x)) = \prod_{q_i} (G_0(x))$ es la probabilidad de que el autómata x tome el estado q_i en el instante inicial ($t = 0$).
- F es la matriz de transición, una matriz de probabilidades de transición entre estados, con dimensión $(\#Q)^{poblacion(A)} \times \#Q \times \#Q$ ($poblacion(A)$ es el número de vecinos y viene determinado por la vecindad V como ya se definió en las secciones precedentes).
  - Para simplificar la notación, se suele considerar F una familia de $(\#Q)^{poblacion(A)}$ matrices cuadradas de dimensión $\#Q \times \#Q$. Se utilizará indistintamente como conjunto de índices para esta matriz cuadrada $[0, .., \#Q - 1] \cap Z$ o directamente Q, según convenga. El estado actual determinará la fila y el estado siguiente la columna dentro de estas matrices.
  - Hay una matriz para cada posible configuración de cualquier vecindario. De manera que si en un momento concreto t_a, para una posición concreta $(\vec{i})$ de la rejilla $C\left(vecindario\left(\vec{i}\right), t_a\right) = \vec{s} = \left(s_0, ..., s_{poblacion(A)-1}\right)$, existe una $F_{\vec{s}}$ (o $F[\vec{s}]$), matriz de dimensión $\#Q \times \#Q$ que contiene las probabilidades de que dada la configuración del vecindario $\vec{s}$, un autómata cualquiera (el valor de $\vec{i}$ es irrelevante, porque F no depende de la posición en la rejilla $\vec{i}$) transite de un estado cualquiera (elemento de Q) a otro. Y en particular $F_{\vec{s},q_i,q_j}$ (o $F[\vec{s}, q_i, q_j]$) será igual al valor de

$$p\left(C\left(G\left[\vec{i}\right], t_a + 1\right) = q_j \middle/ C\left(G\left[\vec{i}\right], t_a\right) = q_i, C\left(G\left[\vec{i}\right], t_a + 1\right) = \vec{s}\right)$$

  - Es decir la probabilidad de que el autómata estudiado en el instante siguiente tome el valor q_j dado que en el instante actual tiene el valor q_i y el vecindario tiene la configuración $\vec{s}$.

4.26 Notación

Dado un autómata celular probabilista $A = (G, G_0, V, Q, F, T)$, resulta evidente que cada autómata finito probabilista a de la rejilla G se deduce de las definiciones anteriores de la siguiente forma:

$$a = (Q^{poblacion(A)}, Q, F, G_0(a), T)$$

4.27 Cálculo del vector de estado de un autómata de la rejilla

4.27.1 Descripción informal

Supóngase un autómata celular probabilista A.

Uno de los problemas relacionados con autómatas celulares probabilistas es la determinación de los vectores de estado de los autómatas de cada posición en cada instante.

Inicialmente la distribución de estados la realiza *inicial* (A) (G_0 en la notación utilizada anteriormente).

Es necesario describir un mecanismo para obtener el vector de estados de cada autómata de la rejilla en los instantes siguientes.

4.27.2 Definición

Sea un autómata celular probabilista A.

Se llama *configuración probabilista de* A, y se escribirá $C^P(A,t)$, o simplemente $C^P(t)$ si no hay posible ambigüedad respecto al autómata considerado, a una función que asigna en cada instante t y a cada elemento de la rejilla de A un vector de estado.

Formalmente,

$C^P(A,t) : rejilla(A) \rightarrow R_{dimension(A)}$ sobre $([0,1] \cap \Re)^{\#estados(A)}$,, $C^P(t)\left[\overrightarrow{i}\right]$ vector de estado de $estados(A)$ para $rejilla(A)\left[\overrightarrow{i}\right]$ en el instante $t\ \forall \overrightarrow{i}$ índice válido sobre $rejilla(A)$.

Se llama *configuración actual probabilista de un autómata probabilista* A y se escribe $C_a^P(A)$ a la que asigna a cada autómata de $rejilla(A)$ su vector de estado actual.

Obsérvese que la configuración probabilista de un autómata define también una configuración probabilista para cualquier subconjunto de la rejilla, que es la restricción al subconjunto de esa configuración.

4.27.3 Algoritmo

Sea un autómata celular probabilista $A = (G, G_0, V, Q, F, T)$.

A continuación se presenta un algoritmo para la determinación de $C^P(A,t)\ \forall t$.

Se define de manera recursiva como sigue:

$$C^P(A,t) =$$

$$\begin{cases} G_0\left(G\left[\vec{i}\right]\right)\forall\vec{i} & si \quad t=0 \\ C^P\left(A,t-1,G\left[\vec{i}\right]\right)\times\left(\sum\left(\prod_x^t C^P(A,t-1,x)\right)\bullet F\right)\forall\vec{i} & si \quad no \end{cases}$$

Donde,

- $\vec{i}$ es un índice válido sobre G.
- $\times$ es el producto de matrices habitual.
- $\prod^t$ es el producto tensorial y, x recorre el conjunto *vecindario* $\left(\vec{i}\right)$, que se ha omitido para hacer más legible la expresión.
- El símbolo del punto "$\bullet$" es el producto elemento a elemento (es decir, se multiplican elementos con el mismo índice) de dos matrices.

En la figura 7.0.e se muestra un esquema gráfico de este proceso.

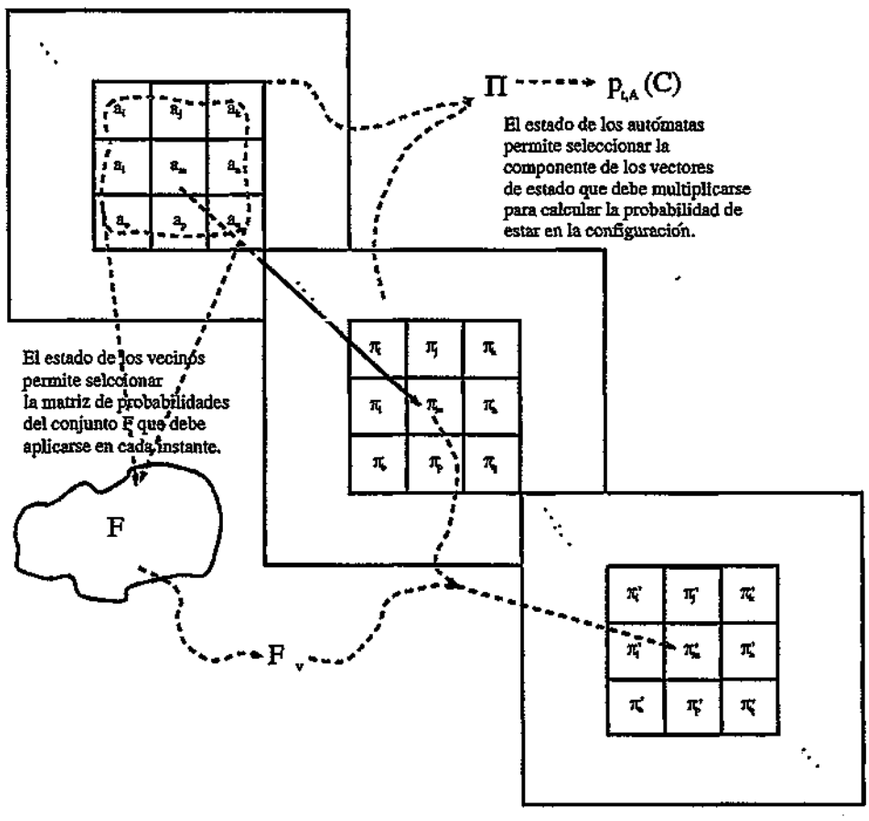


Figura 7.0.e Explicación gráfica del cálculo del vector de estado en $t+1$.

4.28 Ejemplo

El siguiente autómata celular probabilista:

$$pca_1 = \left(\begin{array}{l} R_1 \in R_{\infty,\infty}, \\ G_{0_1}\left(G\left[i,j\right]\right) = \left[p\left(0\right) = 0.5, \quad p\left(1\right) = 0.5\right] \quad \forall i,j, \\ V_N = \left(5, \left(\left(0,0\right), \left(0,1\right), \left(1,0\right), \left(0,-1\right), \left(-1,0\right)\right)\right), \\ Q_1 = \left\{0,1\right\}, \\ F_1, \\ T_1 = \Phi \end{array} \right)$$

Donde:

- R_1 es una rejilla bidimensional infinita.
- La configuración inicial G_{0_1} distribuye estados 0 y 1 de manera equiprobable por la rejilla R_1 .
- Se utiliza la vecindad de von Neumann, que toma 5 vecinos.
- Los autómatas son binarios ($Q_1 = \{0,1\}$).
- F_1 se define de la siguiente manera

$$F_1 = \left\{ \begin{array}{llll} M_{0000} = \begin{bmatrix} 0.1 & 0.9 \\ 0.8 & 0.2 \end{bmatrix} & M_{0001} = \begin{bmatrix} 0.3 & 0.7 \\ 0.4 & 0.6 \end{bmatrix} & M_{0010} = \begin{bmatrix} 0.5 & 0.5 \\ 0.6 & 0.4 \end{bmatrix} & M_{0011} = \begin{bmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{bmatrix} \\ M_{0100} = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix} & M_{0101} = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} & M_{0110} = \begin{bmatrix} 0.5 & 0.5 \\ 0.6 & 0.4 \end{bmatrix} & M_{0111} = \begin{bmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{bmatrix} \\ M_{1000} = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix} & M_{1001} = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} & M_{1010} = \begin{bmatrix} 0.5 & 0.5 \\ 0.6 & 0.4 \end{bmatrix} & M_{1011} = \begin{bmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{bmatrix} \\ M_{1100} = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix} & M_{1101} = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} & M_{1110} = \begin{bmatrix} 0.5 & 0.5 \\ 0.9 & 0.1 \end{bmatrix} & M_{1111} = \begin{bmatrix} 0.2 & 0.8 \\ 0.4 & 0.6 \end{bmatrix} \end{array} \right\}$$

4.28.1 Cálculo del vector de estado

A continuación se muestra un ejemplo del cálculo del vector de estado para un autómata de R_1.

Consideremos la posición (i,j).

Supongamos que en el instante anterior $(t-1)$ los autómatas del vecindario tienen los siguientes vectores de estado:

$$C^P\left(pca_1, t-1, vecindario\left(\left(i,j\right)\right)\right) = \left(\begin{array}{l} \left(0.1, 0.9\right) \\ \left(0.2, 0.8\right) \\ \left(0.6, 0.4\right) \\ \left(0.3, 0.7\right) \\ \left(0.9, 0.1\right) \end{array} \right)$$

Interpretemos la expresión para el cálculo de $C^P\left(pca_1, t, G\left[i,j\right]\right)$ analizada anteriormente

$$C^P\left(pca_1, t-1, G\left[i,j\right]\right) \times \left(\sum \left(\prod_{x \in vecindario\left(\left(i,j\right)\right)}^{t} C^P\left(pca_1, t-1, x\right)\right) \bullet F_1\right)$$

Si la configuración del vecindario del autómata que ocupa la posición (i,j) exlcuyendo el propio autómata fuese $(0,1,0,0)$, la siguiente operación de matrices calcularía el vector de estado buscado:

$$C^P\left(pca_1, t-1, G\left[i,j\right]\right) \times \left(F_1\right)\left[0100\right].$$

Donde "×" representa el producto usual de matrices.

Sin embargo no tenemos seguridad en que la configuración de los demás autómatas del vecindario sea precisamente la elegida $((0,1,0,0))$, aunque sí conocemos la probabilidad de que esto ocurra. Esa probabilidad es el producto de las correspondientes componentes de los vectores de estado de los cuatro vecinos, es decir, la probabilidad de que el vecino superior tome en el instante $t-1$ el estado 0, la probabilidad de que el vecino derecho tome en el instante $t-1$ el estado 1, la probabilidad de que el vecino inferior tome en el instante $t-1$ el estado 0 y la probabilidad de que el vecino izquierdo tome en el instante $t-1$ el estado 0. Se puede usar notación vectorial para acceder a los elementos de $vecindario\,((i,j))$, por tanto

- $vecindario\,((i,j))\,[0]$, es el autómata estudiado: $R_1\,[i,j]$.
- $vecindario\,((i,j))\,[1]$, es el vecino superior: $R_1\,[i,j+1]$.
- $vecindario\,((i,j))\,[2]$, es el vecino derecho: $R_1\,[i+1,j]$.
- $vecindario\,((i,j))\,[3]$, es el vecino inferior: $R_1\,[i,j-1]$.
- $vecindario\,((i,j))\,[4]$, es el vecino izquiero: $R_1\,[i-1,j]$.

$C^P\,(pca_1, t-1)$ proporciona el vector de estado de cada autómata de la rejilla en el instante $t-1$, por tanto

- $C^P\,(pca_1, t-1, vecindario\,((i,j))\,[p])\,,\,p \in \{0,1,2,3,4\}$ son los vectores de estado de los elementos del vecindario.

Y como el vector de estado tiene tantas componentes como elementos hay en Q_1, la primera de ellas para el valor 0 y la segunda para el valor 1, el producto de los siguientes valores expresa la probabilidad buscada:

$$\begin{array}{c} C^P\,(pca_1, t-1, vecindario\,((i,j))\,[1])\,[0] \\ C^P\,(pca_1, t-1, vecindario\,((i,j))\,[2])\,[1] \\ C^P\,(pca_1, t-1, vecindario\,((i,j))\,[3])\,[0] \\ C^P\,(pca_1, t-1, vecindario\,((i,j))\,[4])\,[0] \end{array}$$

Hay una probabilidad para cada una de las posibles configuraciones, será necesario repetir este calculo y sumarlas todas para obtener correctamente el vector de estado del autómata de la posición (i,j). El resultado obtenido es

$$\left(\sum_{i,j,k,l\in\{0,1\}} \left(\begin{array}{c} C^P\,(pca_1, t-1, R_1\,[i,j]) \quad \times \\ \left(\begin{array}{c} C^P\,(pca_1, t-1, vecindario\,((i,j))\,[1])\,[i]\cdot \\ C^P\,(pca_1, t-1, vecindario\,((i,j))\,[2])\,[j]\cdot \\ C^P\,(pca_1, t-1, vecindario\,((i,j))\,[3])\,[k]\cdot \\ C^P\,(pca_1, t-1, vecindario\,((i,j))\,[4])\,[l] \end{array} \right) \cdot \\ (F_1)_{ijkl} \end{array} \right) \right)$$

Donde "$\cdot$" es tanto el producto de números reales como el producto de un número real por una matriz de números reales.

Esta expresión se puede simplificar mediante el producto tensorial tal y como se hizo anteriormente al escribir

$$C^P\,(pca_1, t-1, G\,[i,j]) \times \left(\sum \left(\prod_{x\in vecindario((i,j))}^{t} C^P\,(pca_1, t-1, x)\right) \bullet F_1\right)$$

4.29 Probabilidad de una configuración de un autómata celular probabilista

En la sección anterior se describe cómo se calcula el vector de estado de cada autómata de la rejilla en cada instante.

De lo dicho allí resulta evidente deducir que, exceptuando el vector de estado inicial, en todos los siguientes vectores de estado las probabilidades contenidas en ellos se refieren a que el autómata tome un valor concreto condicionado al valor que toman sus vecinos. Esto ocurre para cada instante y para cada autómata de la rejilla. La vecindad determina los condicionamientos reconocidos por este modelo. Por lo que en cada instante, en cada autómata, los vectores calculados según se especificó en la sección anterior son una función de probabilidad de cada autómata de la rejilla entendido como una variable aleatoria que toma valores del conjunto de estados.

4.29.1 Definición

La probabilidad de que un autómata celular probabilista (A) cuya configuración probabilista es $C^P(A)$ esté en una configuración determinada (C) en un momento determinado (t) se escribirá $p_{t,A}\left(C, C^P(A)\right)$, o simplemente $p(C)$ si no hay ambigüedad posible respecto al resto de elementos de la expresión.

El evento "el autómata A está en la configuración C" puede expresarse como

$\bigcap_{a \in G}$"el autómata a esté en el estado $C(a)$ dado el estado de los demás autómatas de la rejilla"

Como sólo se reconoce el condicionamiento de un autómata por sus vecinos el evento anterior puede expresarse como

$\bigcap_{a \in G}$"el autómata a esté en el estado $C(a)$ dado el estado de los autómatas de su vecindario"

Y los vectores de estado que la configuración probabilista representa contienen las probabilidades de los eventos que son los operandos de la anterior intersección. Como cada componente recoge las probabilidades supuestos todos los condicionamientos permitidos, estos eventos son independientes y el cálculo de sus probabilidades puede realizarse según la siguiente expresión

$$p_{t,A}(C) = \prod_{a \in G} \pi_{C(a)}\left(C^P(a)\right)$$

De las definiciones de las funciones de probabilidad resulta evidente deducir que $p_{t,A}(C)$ también lo es y que cumplirá la propiedad de suma total igual a 1. Formalmente,

$$\sum_{C \in ConjuntoDeTodasLasPosiblesConfiguraciones} p_{t,A}(C) = 1$$

Capítulo 5

Fractales

5.1 Origen de los fractales

En los últimos años del siglo XIX científicos como Cantor, von Koch, Peano y Sierpinski estudiaron algunas curvas con propiedades que las hacían ser consideradas como monstruosas. Las peculiaridades de estos objetos extraños han sido observadas en otros procesos. El análisis de estos fenómenos ha descubierto las características que se han vinculado finalmente con el concepto de fractal.

5.1.1 Modelo *iniciador-iterador*

Algunos de esos procesos pueden describirse mediante una figura inicial (*iniciador*) y una transformación (*iterador*) que convierte esa figura en un conjunto de copias de sí misma de menor tamaño. El iterador vuelve a aplicarse indifinidamente en todas las copias de la figura.

El iniciador del conjunto de Cantor es un segmento cuya longitud se toma como unidad. El iterador elimina un segmento de longitud $\frac{1}{3}$ situado en el centro de cada segmento. A medida que el proceso se repite, el número de segmentos aumenta con el número de aplicaciones del iterador hasta hacerse infinito. Las longitudes, sin embargo, se reducen hasta hacerse 0.

La figura i.f.1 muestra los primeros pasos en la construcción de este conjunto.

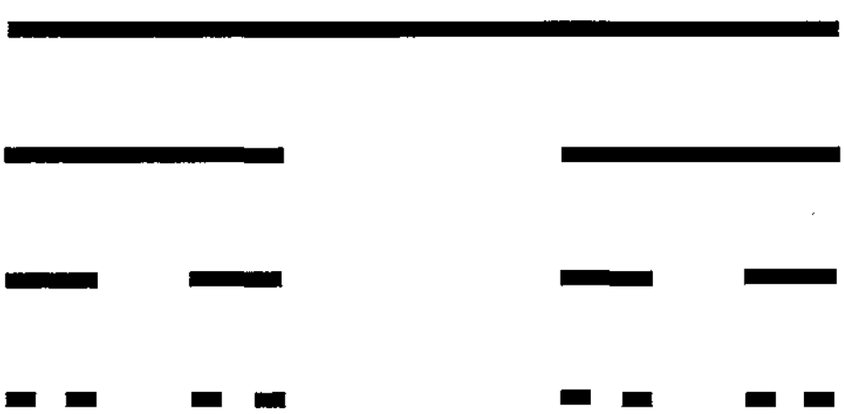


Figura i.f.1: Primeros pasos en la construcción del conjunto de Cantor.

La peculiaridad de este conjunto se observa cuando se intenta medir la longitud de los segmentos que lo componen. Como cada uno de ellos tiene una longitud igual a 0, la suma de todos ellos será también igual a 0. Sin embargo es un conjunto de infinitos puntos y es sorprendente que su *medida* sea nula.

El iniciador básico de la curva de copo de nieve de von Koch es un segmento cuya longitud se toma como unidad. Decimos iniciador básico porque la concatenación de varios segmentos puede formar curvas más complejas y es frecuente iniciar la curva con un triángulo en lugar de con un segmento. El iterador de esta curva sustituye un segmento de longitud $\frac{1}{3}$ situado en el centro de cada segmento por los otros dos lados que completarían con él un triángulo equilatero.

La figura i.f.2 muestra los primeros pasos en la construcción de esta curva.

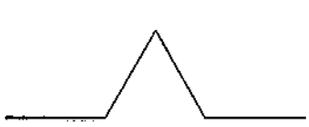


Figura i.f.2: Primeros pasos en la construcción de la curva de von Koch.

El comportamiento anómalo que presenta este caso puede deducirse de la siguiente observación. Imagínese que se fija un punto sobre la curva en cualquiera de sus pasos de construcción. Se considera a ese punto *el punto fijo* y se ata a él un extremo de un trozo de cuerda elástica. Se tira de la cuerda para mantenerla tensa y se fija el otro extremo al final del fragmento de curva dibujado. Si se desplaza este segundo extremo por la curva hasta el principio de la misma y se mantiene la cuerda tirante se apreciará que la inclinación del segmento que forma con el *punto fijo* oscila constantemente. Es decir, no se puede dibujar una tangente constante por *el punto fijo* marcado en la curva. Los puntos con esta propiedad se llaman *loxodrómicos*. Dicho de otra manera, es como si en el límite la curva consistiera en una sucesión de esquinas contiguas, de puntos en los que la tangente a la curva tendría que cambiar de inclinación respecto a sus vecinos. De este hecho se deducen las dos características del comportamiento anómalo: la curva no es derivable en ninguno de sus puntos y además es un trazo contínuo.

Para comprender mejor el significado de la última de las propiedades puede intentarse calcular la longitud recorrida por la curva. En el paso enésimo la curva consiste en 4^n segmentos de longitud $\frac{1}{3^n}$. Por lo tanto la longitud total es $\left(\frac{4}{3}\right)^n$. Por expresarlo de una manera intuitiva, el número de segmentos crece más de lo que disminuye su longitud. Eso significa que la longitud total cuando n tiende a infinito también tiende a infinito como puede observarse en la siguiente tabla que aparece en [*Flk*98]

Paso	N? segmentos	Longitud segmento	Longitud total
0	1	1	1
1	4	0.33333333	1.33333333
2	16	0.11111111	1.7777778
3	64	0.037037037	2.37037
4	256	0.0123457	3.16049
5	1024	0.00411523	4.21399
6	4096	0.00137174	5.61866
7	16384	0.000457247	7.49154
8	65536	0.000152416	9.98872
9	262144	5.08053×10^{-5}	13.3183
⋮	⋮	⋮	⋮
100	1.60694×10^{60}	1.94033×10^{-48}	3.11798×10^{12}

Obsérvese que los extremos del fragmento de curva representado no cambian según se avanza en el proceso de construcción de la misma. Toda la curva está dentro de un rectángulo de base igual a 1 y altura igual a $\frac{1}{3}$. Es sorprendente que un fragmento de curva acotada tenga

una longitud infinita. También es sorprendente que la curva no tenga discontinuidades sin ser derivable en ningún punto.

El iniciador de la curva de Peano es un segmento cuya longitud se toma como unidad. El iterador de esta curva sustituye un segmento de longitud $\frac{1}{3}$ situado en el centro de cada segmento por un ocho centrado en el segmento formado por dos cuadrados de lado igual a $\frac{1}{3}$ de manera que la base de uno de ellos coincide con el segmento central y con el lado superior del otro cuadrado. La figura i.f.2 muestra con detalle este iterador. Los números y direcciones de los segmentos indican cómo se realiza el trazo sin levantar el lápiz del papel.

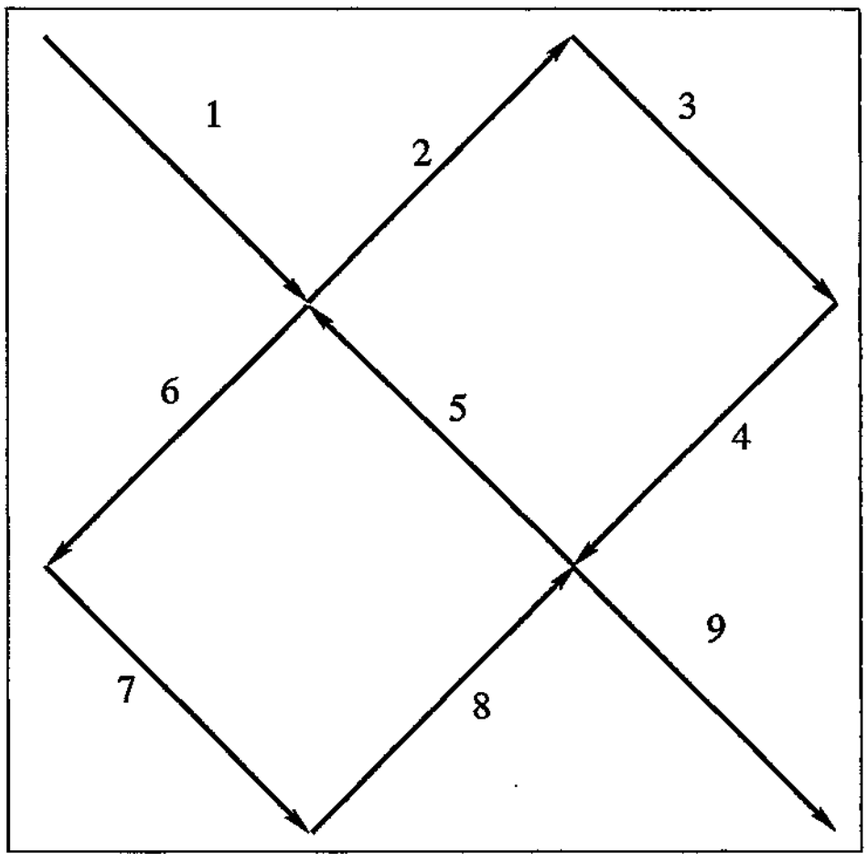


Figura i.f.2: Iterador de la curva de Peano.

Las figuras i.f.3 e i.f.4 muestran los siguientes pasos de construcción de esta curva.

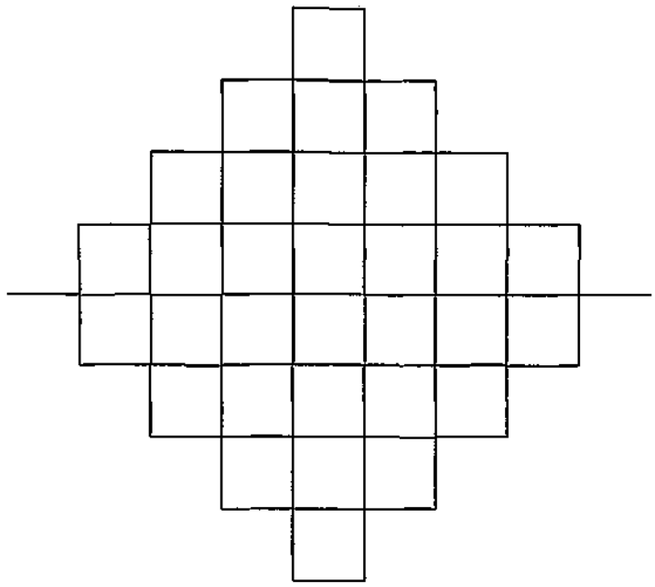


Figura i.f.3: siguiente paso de construcción de la curva de Peano.

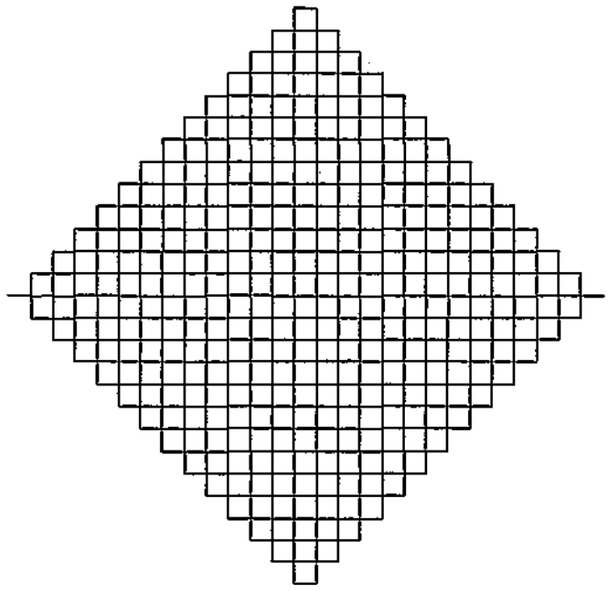


Figura i.f.4: siguiente paso de construcción de la curva de Peano.

La curva de Peano, al igual que la de von Koch, tiene una longitud infinita en una región acotada. Además, en el límite, pasa por todos los puntos de la región plana que la acota. Es curioso que una curva (el recorrido de la curva de Peano puede realizarse sin levantar el lápiz del papel) sea capaz de pasar por todos los puntos de una región plana.

5.1.2 Comportamientos fractales aleatorios

Algunos procesos estocásticos tienen la propiedad de ser parecidos a sí mismos en diferentes escalas. Esto quiere decir que si se estudia dos representaciones gráficas de esos procesos eliminando las referencias de escalas espaciales y temporales es difícil determinar las relaciones temporales o espaciales entre ellas.

Uno de estos procesos es el movimento de las partículas suspendidas en líquidos conocido como movimiento Browniano. Al estudiar la posición de una partícula suspendida en un fluido en intervalos de tiempo regulares se observa un comportamiento similar al que se obtendría al generar parejas aleatorias de incrementos de coordenada x e y ($\{(\Delta x_i, \Delta y_i)\}_{i=1}^{\infty}$) y representar, a partir de una posición inicial (x_0, y_0), la secuencia de posiciones $\{(x_0 + \Delta x_i, y_0 + \Delta y_i)\}_{i=1}^{\infty}$.

La figura i.f.5 muestra un ejemplo de movimiento browniano.

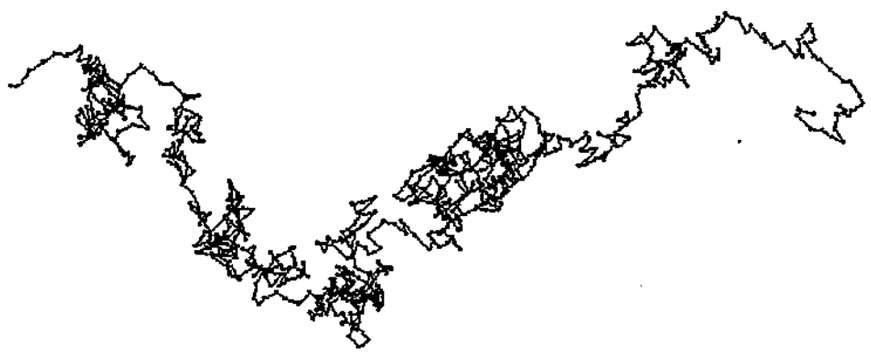


Figura i.f.5: ejemplo de camino aleatorio, movimiento Browniano.

5.1.3 Autosemejanza por transformaciones afines

Algunas imágenes naturales o sintéticas se pueden describir como un conjunto de copias de un patrón sutilmente modificadas. La modificación puede consistir en que en la copia, la imagen original aparezca girada, con un tamaño distinto o simplemente desplazada respecto a la posición de las demás. Algunos algoritmos han sido diseñados para construir imágenes de este tipo. Uno de ellos es el MRCM (***M***ultiple ***R***eduction ***C***opy ***M***achine). Este algoritmo construye un gráfico aplicando indefinidamente un proceso que consiste en distribuir un número determinado de imágenes obtenidas al someter una imagen patrón a alguna transformación geométrica afín (combinación lineal de giros, cambios de escala y traslaciones). Es importante señalar que tanto el número de copias como las transformaciones aplicadas no cambian en todo el proceso.

La figura i.f.6 muestra un esquema del algoritmo MRCM.

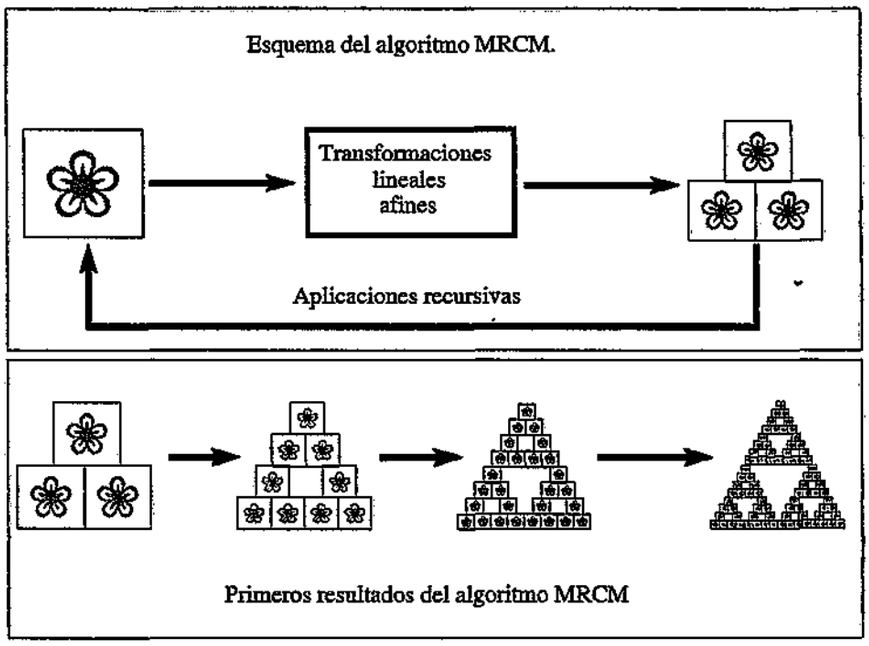


Figura i.f.6: esquema del algoritmo MRCM.

El resultado del algoritmo MRCM se obtiene cuando el número de repeticiones del proceso tiende a infinito, por lo tanto, el número de transformaciones afines del patrón se incrementa exponencialmente. La técnica conocida como IFS (*I*terated *F*unction *S*ystem) y debida a Michael Barnsley [*Bar*88] resuelve los problemas de eficiencia del algorimo MRCM. IFS aprovecha las propiedades de las transformaciones afines para, sustituir el tratamiento completo del patrón por el tratamiento aleatoriamente parcial de algunos puntos elegidos al azar. De esta forma consigue mostrar un subconjunto de puntos más cercano al resultado final. La imagen obtenida tiene la suficiente información como para que la aproximación resulte aceptable.

La figura i.f.7 muestra un esquema de la técnica IFS.

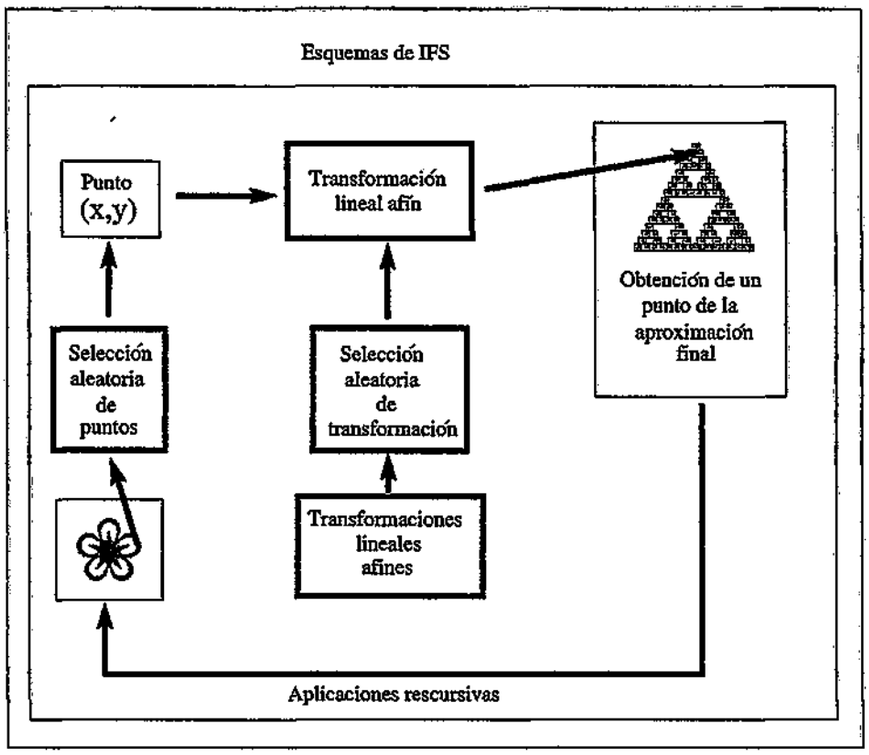


Figura i.f.7: esquema de la técnica IFS.

5.1.4 Sistemas dinámicos iterativos

Los conjuntos de Mandelbrot y Julia se han hecho famosos en el último cuarto de siglo. Provienen del estudio del dominio de convergencia de funciones complejas de variable compleja iterativas. Estas funciones se aplican indefinidamente sobre el resultado de aplicarse ellas mismas a partir de un valor inicial.

En estos conjuntos pueden apreciarse los mismos fenómenos observados en casos anteriores: complejidad independiente de las escalas utilizadas, repetición de las mismas estructuras a lo largo de todo el conjunto, etc.

El conjunto de Mandelbrot estudia el dominio de convergencia del siguiente sistema dinámico iterativo:

- Se selecciona una región (una bola) del plano complejo.
- Para cada elemento c de esa región se realiza el siguiente tratamiento:
  - Se toma como valor inicial $x_0 = 0 + 0i$.
  - Se calcula $x_{t+1} = f(x_t) = x_t^2 + c$ hasta determinar si para ese valor c el proceso converge o diverge.

Se puede realizar simplificaciones para que la determinación de la divergencia resulte eficiente.

Al investigar una bola rectangular cuya esquina superior izquierda es $-2.4 + 1.4i$ y cuya esquina inferior derecha es $1.34 - 1.4i$ Mandelbrot descubrió la silueta de la visión general del conjunto que lleva su nombre.

El conjunto de Julia está intimamente relacionado con el de Mandelbrot. Analíticamente la única diferencia con el proceso descrito anteriormente es que ahora se escoge un valor constante c y el parámetro que recorre la región del plano complejo seleccionada es el valor inicial x_0.

El proceso completo puede resumirse de la siguiente manera:

- Se selecciona una región (una bola) del plano complejo.
- Se selecciona un valor constante complejo c.
- Para cada elemento x_0 de la región seleccionada se realiza el siguiente tratamiento
  - Se calcula $x_{t+1} = f(x_t) = x_t^2 + c$ hasta determinar si para ese valor x_0 el proceso converge o diverge.

Se ha estudiado exhaustivamente las relaciones entre ambos conjuntos de forma que analizando el comportamiento de c en el conjunto de Mandelbrot puede deducirse las propiedades que tendrá el conjunto de Julia para ese valor de c.

5.2 El concepto de fractal

En la sección anterior se ha presentado ciertas propiedades comunes en todos los ejemplos:

- Identificación de los mismos patrones a lo largo de toda la estructura.
- Comportamientos anómalos como, por ejemplo, conjuntos de infinitos puntos de "medida" nula, curvas acotadas de longitud infinita, curvas que recorren el plano por completo, etc...

La primera de las propiedades se asocia al término auto-semejanza. La segunda ha motivado la revisión del concepto topológico de dimensión.

Algunos de los ejemplos anteriores muestran además otras características como, por ejemplo, la no derivabilidad en ningún punto de la curva.

Benoît Mandelbrot acuñó el término fractal para englobar a todos estos fenómenos que, por cumplir todas o algunas de las propiedades identificadas, habían sido excluídos de una clasificación formal. Su definición se redujo a la comparación entre dos medidas de la dimensión. La clásica, a la que dio el nombre de topológica, y una nueva medida de la dimensión, que llamó fractal. Los objetos fractales tienen dimensión fractal estrictamente mayor que la topológica. En el caso de los objetos estudiados por la geometría clásica, ambas cantidades coinciden.

Tras esta definición los cuatro grupos de casos analizados anteriormente dieron origen a cuatro grandes clases de fractales que abarcan la mayoría de los fenómenos de este tipo.

- Fractales de tipo *iniciador - iterador* (como, por ejemplo, las curvas de von Koch, de Peano, conjunto de Cantor).
- Fractales aleatorios (como, por ejemplo, el movimiento browniano, el ruido blanco, las trayectorias estocásticas).
- Fractales auto-semejantes por afinidad, o auto-afines (como, por ejemplo, los generados por el algoritmo MRCM/IFS).
- Fractales de sistemas dinámicos iterativos (como, por ejemplo, conjuntos de Mandelbrot, Julia).

Todos los trabajos presentados en esta tesis relativos a fractales tienen que ver con los fractales del primer grupo, con los problemas de su representación mediante sistemas de Lindenmayer y con el cálculo de su dimensión fractal.

5.3 La representación de los fractales de tipo iniciador-iterador mediante sistemas de Lindenmayer

Se ha encontrado inspiración en la naturaleza para resolver el problema de la representación compacta y sencilla de estructuras de enorme complejidad. Las formas de los plieges del cerebro, las ramificaciones del sistema vascular y respiratorio, presentan las mismas propiedades que se han analizado en los ejemplos anteriores de fractales. Toda esa infinita complejidad está codificada de una manera compacta en el cargamento genético de los núcleos de las células. Las gramáticas formales son, en cierto sentido, una manera compacta de codificar los lenguajes, que son conjuntos de considerable complejidad. En particular, los gramáticas formales de derivación paralela o esquemas de Lindenmayer, que ya se han presentado en el capítulo 2 , han tenido éxito en la representación del crecimiento de muchos organismos, en la descripción de la topología de redes neuronales capaces de resolver determinados problemas [*Brs*92] y en la representación de algunos fractales.

Desde su aparición como formalismo de codificación de las reglas del desarrollo y crecimiento de organismos pluricelulares, los sistemas L han sido utilizados también para representar los fractales expresados mediante una pareja *iniciador-iterador*. Los detalles de esta representación se analizan con más detenimiento en el capítulo 6. La clave para ello consiste en asociar una palabra (axioma) al iniciador, y reglas de producción al iterador, además de utilizar una interpretación gráfica adecuada que permita obtener imágenes al interpretar las palabras derivadas por la gramática.

5.4 El problema de la dimensión fractal.

El concepto de dimensión es muy antiguo y parece sencillo y evidente. Vivimos en un espacio de tres dimensiones: longitud, anchura y profundidad. Algunos objetos en nuestro entorno son más o menos bidimensionales: una hoja de papel, un cuadro, el tablero de una mesa.... otros tienen dimensión uno: un tramo de carretera sobre un plano, el trazo de un lápiz en un papel.

Lo que llamamos dimensión está intuitivamente asociado al número de direcciones en las que se permite el movimiento, es decir, el número de grados de libertad.

Formalmente se suele asignar dimensión 0 a los puntos, 1 a las curvas, 2 a las superficies, 3 a los volúmenes, etc... Esta secuencia de enteros consecutivos da una solución que parece muy clara y elegante al problema de la dimensión. Sin embargo, como Mandelbrot demostró en su famoso libro sobre fractales, [*Man75*] hay objetos excepcionales que son casos dudosos en los que no está claro qué entero asignar como dimensión. Para presentar estas dificultades, se realiza la siguiente reflexión respecto a la dimensión de un ovillo de hilo percibida por observadores de distinto tamaño y separados por distintas distancias:

- Si el tamaño del observador es muy grande (grande como una montaña o un planeta) el ovillo parecerá un insignificante punto. Por tanto podrá asignársele 0 como dimensión.
- Si el observador tiene un tamaño comparable al del ovillo y se encuentra separado de él cierta distancia, éste parecerá una bola (o una esfera). Por tanto se le podrá asignar 3 como dimensión.
- Si el observador tiene un tamaño menor (una hormiga, por ejemplo) y están cerca, su percepción del ovillo será la de una línea enrollada. Como toda curva, tendrá dimensión 1.
- Si el observador es más pequeño que el ovillo, de un tamaño comparable al grosor del hilo (una bacteria, por ejemplo), su percepción será la de un cilindro enrollado. Es decir, un volumen al que corresponderá dimensión 3.
- Si el tamaño del observador es aún menor, tanto como para percibir los átomos como minúsculos puntos en el espacio, el ovillo también será percibido como un conjunto de puntos aislados, por tanto le corresponderá dimensión 0.
- Observadores menores, de tamaño comparable con los átomos, percibirán el entorno (incluido el ovillo) como un conjunto de esferas, es decir, volúmenes de dimensión 3.

El concepto de dimensión presenta problemas que no se esaperaban *a prirori.*

La definición de fractal realizada por Mandelbrot parece ser una solución definitiva. Pero el concepto de dimensión fractal resulta un tanto vago y esquivo. En la actualidad coexisten distintas medidas de la dimensión fractal que coinciden en su propósito e incluso en los valores que calculan para la mayoría de los casos, pero que difieren en matices que hacen que unas sean más adecuadas que otras en determinadas circunstancias.

Mandelbrot recogió los trabajos del meteorólogo Lewis Richardson en sus estudios sobre la longitud de las costas y fronteras de diferentes países. Richardson identificó el problema de la dependencia entre el resultado obtenido y el tamaño del patrón unidad utilizado para medir. Este fenómeno de las costas y las fronteras se ha comentado ya en los ejemplos de fractales del principio del capítulo. Si se marca dos puntos sobre la línea de costa (o la frontera entre dos países) en un mapa y se mide la distancia entre los dos puntos con un compás se obtiene una medida de la longitud entre los dos puntos. Si, manteniendo los puntos marcados, se repite la operación sobre un mapa con escala mayor, los tramos aparentemente rectilíneos con poco

detalle se convertirán en líneas sinuosas que recorren accidentes inapreciables con menor detalle y el resultado obtenido será dramáticamente distinto.

Richardson encontró que la relación entre los logaritmos de la longitud medida y la unidad de medida utilizada definía con bastante precisión una línea recta. Consideró que la pendiente de esa recta era un buen descriptor de las curvas que estaba estudiando. A esa pendiente le asociaría Mandelbrot el concepto de dimensión fractal.

Formalmente:

- Sea p_l la longitud del patrón.
- Se llamará $N(p_l)$ al número de pasos de longitud p_l necesario para recorrer la curva estudiada.
- La *dimensión de Richardson-Mandelbrot* D_p se define como el siguiente límite

$$D_p = \lim_{p_l \to o^+} \left(\frac{-\log(N(p_l))}{\log(p_l)} \right)$$

Obsérvese la generalización que la dimensión de Richardson-Mandelbrot supone respecto a la dimensión de la geometría clásica. Cuando se mide un objeto de los estudiados por la geometría clásica (una curva, una superficie), entre $N(p_l)$ y p_l hay siempre una relación constante:

$$N(p_l) = \left(\frac{1}{p_l} \right)^D$$

Donde en este caso D es la dimensión topológica de la geometría clásica.

Esta expresión es otra manera de decir que la medida de las rectas es lineal respecto a p_l (con $D = 1$ y entendiendo medida como longitud), la medida de las figuras planas es cuadrática respecto a p_l (con $D = 2$ y entendiendo medida como superficie), la medida de los cuerpos es cúbica respecto a p_l (con $D = 3$ y entendiendo medida como volumen). Manipulando esa expresión se puede obtener fácilmente el cociente cuyo límite define la dimensión de Richardson-Mandelbrot.

Con esta definición Mandelbrot consiguió, según sus palabras, *domesticar* casos de objetos monstruosos, obteniendo dimensiones fractales con un significado intuitivamente correcto:

- El conjunto de Cantor tiene una dimensión fractal de Richardson-Mandelbrot $D_p = \frac{\log(2)}{\log(3)} \simeq 0.63093$, que tiene el significado intuitivo siguiente: los elementos del conjunto de Cantor son infinitos puntos aislados, que no llegan a comportarse como una curva (de dimensión 1), pero su naturaleza es más compleja que la de un simple punto (dimensión 0).
- La curva copo de nieve de von Koch tiene una dimensión fractal de Richardson-Mandelbrot $D_p = \frac{\log(4)}{\log(3)} \simeq 1.26186$, que tiene el significado intuitivo siguiente: la curva de von Koch tiene longitud inifinita, pero dentro de una banda de ancho limitado, es decir, su naturaleza es más compleja que la de una curva, de longitud finita en una banda acotada (de dimensión 1) pero no llega a comportarse como una superficie o un plano (de dimensión 2).
- La curva de Peano tiene una dimensión fractal de Richardson-Mandelbrot $D_p = 2$ que

tiene un significado intuitivo evidente: realmente la curva de Peano, al conseguir cubrir el plano, se comporta como una superficie (de dimensión 2).

La de Richardson-Mandelbrot no es la única posible dimensión fractal. Entre otras se puede mencionar las dimensiones de Hausdorff, de Hausdorff-Besicovich, de Minkowski y la dimensión de *contar cajas*.

Parte II

Estudio de fractales mediante sistemas de Lindenmayer

Capítulo 6

Clasificación gráfica de los sistemas L para representar fractales

6.1 Fractales y sistemas de Lindenmayer

Ya se ha presentado en el capítulo 5 el uso de sistemas de Lindenmayer para representar fractales del tipo *iniciador-iterador* y la necesidad de añadir a los sistemas L interpretaciones gráficas de las cadenas de sus lenguajes para que esto sea posible.

Es necesario distinguir claramente entre el sistema L y la interpretación gráfica. La confusión es frecuente en la literatura y lleva a que deficiencias en la interpretación gráfica se intenten resolver en el sistema L y viceversa, motivando a veces extensiones a unos y otros que realmente no son necesarias ([*Gie*91], [*Pru*86]).

Se ha utilizado dos familias diferentes de interpretaciones gráficas para los sistemas de LIndenmayer: la interpretación gráfica tortuga y la vectorial. Las dos son descritas a continuación.

Gráficos tortuga

Los gráficos tortuga fueron creados en 1980 por Papert [*Pap*80]. Un gráfico tortuga es el rastro dejado por una tortuga invisible cuyo estado en cada instante es definido por su posición y la dirección a la que apunta. El estado de la tortuga cambia a medida que se desplaza un paso hacia delante una longitud determinada y siempre la misma m o rota un ángulo determinado y siempre el mismo α sin modificar su posición.

Las interpretaciones gráficas tortuga pueden mostrar diferentes niveles de complejidad. El más sencillo de ellos considera que el alfabeto de los sistemas de Lindenmayer contiene sólo los tres símbolos siguientes:

$$\Sigma = \{F, +, -\}$$

Su interpretación gráfica se describre a continuación

F La tortuga se mueve un paso hace delante, en la dirección y sentido a los que apunta, dejando un rastro visible. Los símbolos que implican un desplazamiento que deja un rastro visible se llaman *símbolos de* dibujo.

$+$ La tortuga gira un ángulo positivo igual a α.

$-$ La tortuga gira un ángulo negativo igual a α.

En el próximo capítulo se muestran ejemplos de sistemas L con esta interpretación gráfica.

A estos símbolos básicos se añade reglas adicionales que complican los gráficos tortuga y que posibilitan representar fractales de diferentes familias. A continuación se muestra algunas extensiones frecuentes:

- Las letras mayúsculas distintas de F no tienen representación gráfica y dejan el estado de la tortuga inalterado. Llamaremos *símbolos no gráficos* a este tipo de símbolos.
- La letra efe minúscula (f) hace que la tortuga avance un paso de longitud m hacia delante sin dejar rastro visible. Llamaremos *símbolos de movimiento* a este tipo de símbolos.
- El símbolo de paréntesis de apertura ((), guarda el estado de la tortuga en una pila. El símbolo de paréntesis de cierre ()) saca de la pila el último elemento guardado que se convierte en el estado actual de la tortuga. Esta extensión permite representar fractales ramificados. En la presente tesis sólo serán consideradas cadenas que tengan los paréntesis balanceados de acuerdo a las reglas sintácticas usuales ya que, por ejemplo, un paréntesis de cierre sin un paréntesis de apertura previo causaría un error debido a que la pila está vacía.
- Puede añadirse, sin restricciones, símbolos adicionales no gráficos y de movimiento.
- El símbolo de cierre de admiración (!) hace que la tortuga gire 180 grados.
- Se utiliza las llaves ({y}) para indicar que el área encerrada entre ellas se rellene con algún color.
- Puede añadirse símbolos adicionales para manipular el color de los rastros visibles de los símbolos de dibujo.

A lo largo de esta tesis, se prestará atención a todas las extensiones excepto a las tres últimas.

Los gráficos de tortuga son muy flexibles. Admiten múltiples extensiones que aumentan su potencia expresiva. Sin embargo, su representación es inherentemente lenta ya que el estado de la tortuga en cualquier punto es una función no trivial de la historia completa de los movimientos previos. No queda más remedio que calcular los pixels de la representación gráfica de las cadenas de forma secuencial por medio de un bucle complicado.

Gráficos vectoriales

En esta familia de interpretaciones gráficas, cada símbolo del alfabeto del sistema L es asociado a un vector en el sistema cartesiano rectangular. Una palabra es representada por la concatenación de los vectores de los símbolos que componen la palabra.

Esta interpretación permite representar fractales ramificados ya que siempre es posible volver al inicio de la rama si para cada símbolo en el alfabeto existe otro símbolo asociado al vector opuesto.

Es necesario extender esta representación para construir fractales con discontinuidades como por ejemplo el conjunto de Cantor. Para ello, en lugar de asociar a cada símbolo del alfabeto un par con las coordenadas x e y del extremo del vector, se le asocia una terna que añade a estas coordenadas una componente binaria de manera que el valor 1 indica que el vector es visible y el valor 0 indica que el vector es invisible.

Los gráficos vectoriales son menos flexibles que los graficos tortuga. No es fácil extenderlos, por ejemplo, para el relleno de áreas cerradas. Sin embargo, la composición de vectores es una operación muy sencilla que puede implementarse mediante un bucle muy sencillo lo que quiere decir que habitualmente los gráficos vectoriales son más eficientes que los gráficos tortuga.

En las próximas secciones se define la notación que será utilizada a lo largo de la tesis para referirse a estos conceptos.

6.2 Definición, vector opuesto de un vector dado.

Entenderemos por *vector opuesto* lo mismo que significa para la operación suma de espacios vectoriales.

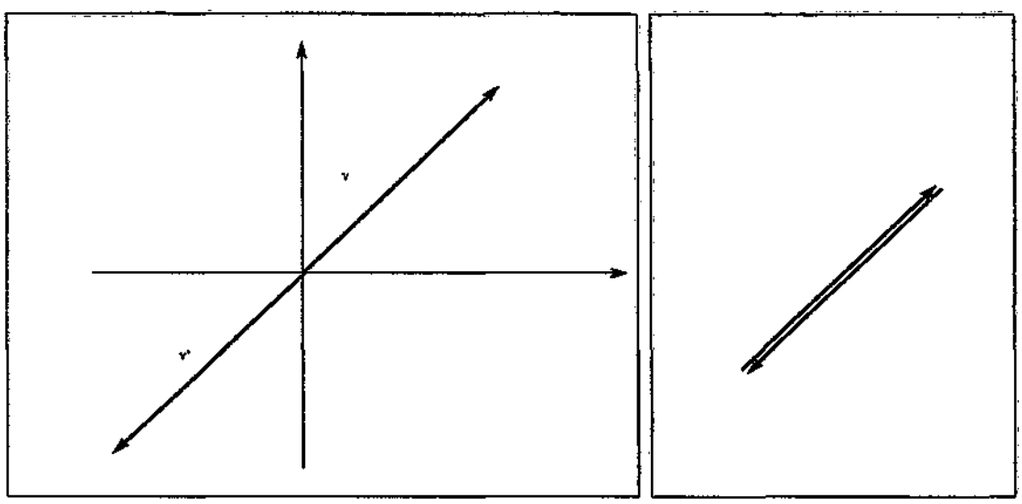
Formalmente:

$$\forall \vec{v} \text{ (vector) } \vec{v'} \text{ (también vector) es opuesto de v} \Longleftrightarrow \vec{v} + \vec{v'} = \vec{v'} + \vec{v} = \vec{0}$$

Donde $\vec{0}$ es el elemento neutro de la suma de vectores:

$$\forall \vec{v} \text{ (vector) } \vec{v} + \vec{0} = \vec{0} + \vec{v} = \vec{v}.$$

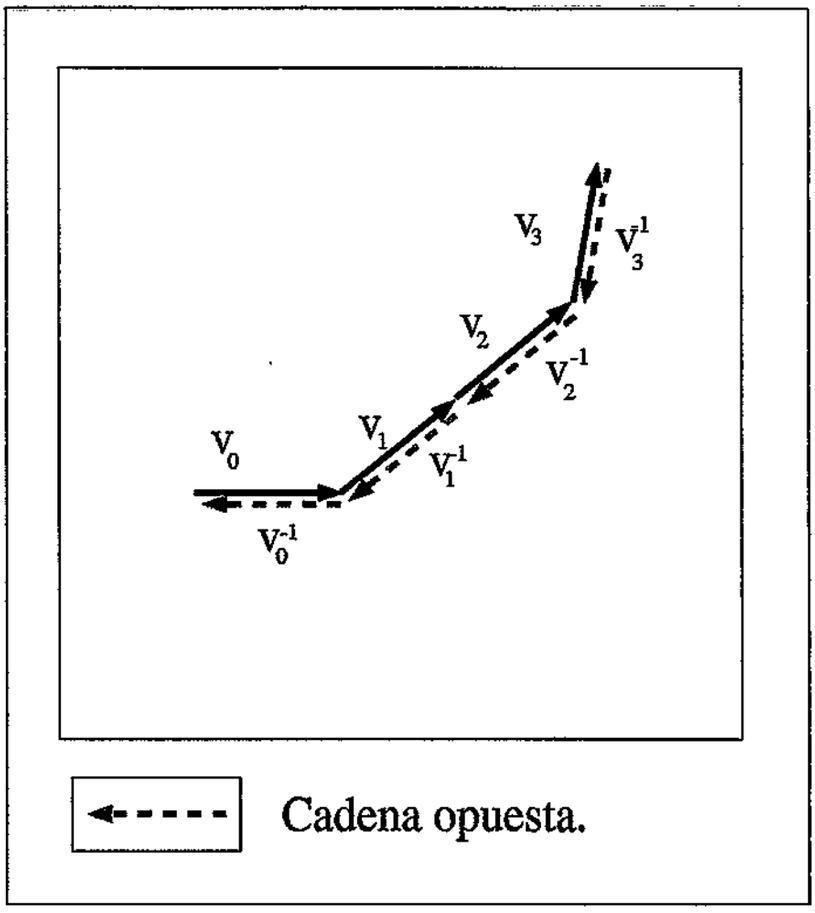
Gráficamente. La concatenación de los vectores lleva al origen del primero.



6.3 Definición, cadena de vectores opuesta de una cadena dada.

Dada una cadena $x = \overrightarrow{v_0}\overrightarrow{v_1}...\overrightarrow{v_p}$ se define la cadena opuesta de x y se llama x^{-1} a la cadena $x^{-1} = \overrightarrow{v_p}^{-1}\overrightarrow{v_{p-1}}^{-1}...\overrightarrow{v_0}^{-1}$ donde cada $\overrightarrow{v_i}^{-1}$es el vector opuesto de $\overrightarrow{v_i}$.

En la siguiente gráfica se muestra cómo $(\overrightarrow{v_0}\overrightarrow{v_1}\overrightarrow{v_2}\overrightarrow{v_3})^{-1} = \overrightarrow{v_3}^{-1}\overrightarrow{v_2}^{-1}\overrightarrow{v_1}^{-1}\overrightarrow{v_0}^{-1}$.



6.4 Teoría de grupos e interpretaciones gráficas

Algunos resultados de teoría de grupos serán utilizados a lo largo de la siguiente tesis. Serán enunciados donde convenga y, si es necesario, las demostraciones, que pueden ser encontrados en textos básicos de álgebra de teoría de grupos, serán sustituídas por justificaciones informales.

6.5 Definición, interpretación gráfica de tortuga (G_T) de un sistema L tortuga (S).

Sea Σ el alfabeto del sistema S.

Se define formalmente interpretación tortuga como la quíntupla siguiente.

$$G_T = (D, M, U, \alpha, m)\, D, M, U \subseteq \Sigma; \alpha = \tfrac{2k\pi}{n} \in \Re, , n, k \in Z \wedge m \in \Re$$

Donde:

- $D, M, U \subseteq \Sigma$, son subconjuntos del alfabeto y hacen referencia respectivamente:
  - D conjunto de símbolos de dibujo. Suele contener, al menos, al conjunto de símbolos $\{F\}$
  - M conjunto de símbolos de movimiento. Suele contener, al menos, el conjunto de símbolos $\{f\}$
  - U conjunto de símbolos ignorados por la representación gráfica.
  - Σ al menos contiene, en general, el conjunto de símbolos $\{F, f, +, -, (,)\}$
- α es el ángulo básico de la interpretación tortuga.
- m es el módulo del desplazamiento básico de la interpretación tortuga.

A lo largo de este trabajo se utilizará la siguiente notación para referirse a las correspondientes componentes de una interpretación tortuga G_T:

- ángulo$(G_T) = \alpha$.
- módulo$(G_T) = m$.
- alfabeto_dibujable$(G_T) = D$.
- alfabeto_movimiento$(G_T) = M$.
- alfabeto_no_gráfico$(G_T) = U$.
- alfabetos$(G_T) = D \cup M \cup U$.

6.6 Teorema

Los posibles incrementos de ángulo expresables en una interpretación gráfica de tortuga con $\alpha = \frac{2k\pi}{n}$ son nZ (el grupo cíclico {0,...,n-1})

6.6.1 Justificación informal

La naturaleza cíclica de los posibles ángulos con estas características $(i\alpha, , \alpha = \frac{2k\pi}{n} \wedge i \in Z)$ y del grupo cíclico formado por la relación *mismo resto al dividir entre* n en el conjunto $\hat{Z}$ $(\{0, ..., n-1\})$, es la clave de la demostración de este resultado.

La demostración formal se basa en resultados como los siguientes [*Que*79]:

- Si n es un entero estrictamente positivo, el grupo de las rotaciones del plano de centro O y de ángulo $\frac{2k\pi}{n}$ (con k entero racional cualquiera) es isomorfo al grupo aditivo Z/nZ.
- Todo grupo monógeno de orden n es isomorfo a Z/nZ.

6.7 Definición

El *conjunto de ángulos asociados a una interpretación gráfica tortuga con* $\alpha = \frac{2k\pi}{n}$ se define según la siguiente expresión:

$$\Theta\left(G_T\right) = \{i\alpha\}_{i \in nZ}$$

6.8 Definición

El *conjunto de vectores asociados a una interpretación gráfica tortuga con* $\alpha = \frac{2k\pi}{n}$ se define según la siguiente expresión:

$$V\left(G_T\right) = \{(m,\beta),, m = \text{módulo}(G_T) = m; \beta \in \Theta\left(G_T\right)\}$$

6.9 Definición, compatibilidad entre gráficos tortuga y sistemas de Lindenmayer

Se dice que una *interpretación gráfica de tortuga* $G_T = (D, M, U, \alpha, m)$ es *compatible con un sistema de Lindenmayer* $S \Leftrightarrow D \cup M \cup U \subseteq alfabeto\left(S\right) \subseteq D \cup M \cup U \cup \{+, -, (,)\}$

6.10 Definición, gráfico de Lindenmayer

Se llama *gráfico de Lindenmayer* al par $G_L = (L, G)$ donde L es un sistema de Lindenmayer, G es una interpretación gráfica vectorial o tortuga compatible con L.

Si se necesita hacer referencia a las partes del gráfico de Lindenmayer se utlizará la notación siguiente:

- $sistema(G_L)$ representará el sistema de Lindenmayer del gráfico G_L.
- interpretación(G_L) representará la interpretación gráfica del gráfico G_L.

De forma que se cumplirá la siguiente condición:

$$G_L = \left(sistema\left(G_L\right), \text{interpretación}\left(G_L\right)\right)$$

6.11 Definición, gráfico tortuga de Lindenmayer

Se llama *gráfico tortuga de Lindenmayer* a aquél cuya interpretación gráfica es de tipo tortuga.

6.12 Nomenclatura relacionada con los gráficos de Lindenmayer.

A lo largo de este trabajo, se utilizará respecto a los gráficos de Lindenmayer las mismas expresiones utilizadas para referirse a los sistemas de Lindenmayer y a las interpretaciones gráficas. Así, por ejemplo, cuando se diga la interpretación gráfica de una cadena derivada a partir de otra por un gráfico de Lindenmayer en realidad se estará haciendo referencia al resultado de aplicar la interpretación gráfica del gráfico de Lindenmayer a la cadena derivada a partir de otra por el sistema L del gráfico de Lindenmayer.

6.13 Sistemas L gráficamente equivalentes.

6.13.1 Definición.

Dos sistemas L son gráficamente equivalentes si representan la misma curva fractal por medio de alguna interpretación gráfica. Formalmente lo son si para cada uno de ellos existe una interpretación gráfica tal que los gráficos de Lindenmayer formados por los sistemas y las interpretaciones representan el mismo fractal.

6.14 Esquemas L gráficamente equivalentes.

6.14.1 Definición.

Dos esquemas L son gráficamente equivalentes si para cada sistema L del primero (es decir, para cada posible axioma que se añada al primer esquema para formar un sistema L), existe un sistema L gráficamente equivalente en el segundo.

6.14.2 Observación.

Dos sistemas L pueden ser gráficamente equivalentes y sus esquemas L no tienen necesariamente que serlo.

Dos sistemas D0L pueden ser gráficamente equivalentes independientemente de su clase, es decir, ambos pueden ser TGD0L o ambos pueden ser VGD0L o de un tipo distinto cada uno de ellos.

6.15 Cadenas invariantes respecto al ángulo de sistemas L con interpretación gráfica de tipo tortuga

6.15.1 Definición

Se llama *inclinación de la tortuga en la interpretación gráfica tortuga* (y se escribe φ) al ángulo que forma con el eje de abscisas el segmento dibujado al interpretar un símbolo de dibujo en la

posición actual de la tortuga.

6.15.2 Definición

Suponiendo un origen de desplazamientos (x_0, y_0) (usualmente (0,0)) y de inclinaciones α_0 (usualmente 0), se llama *estado de la tortuga en la interpretación gráfica tortuga* al par $((x,y),\varphi)$ donde (x,y) es la posición (coordenadas cartesianas) de la tortuga y φ su inclinación.

6.15.3 Definición,

Se llama *estado de la tortuga asociado a una cadena* u al estado de la tortuga tras la interpretación gráfica de la misma.

Si dicho estado es $((x,y),\varphi)$ se llamará respectivamente *posición* e *inclinación asociadas a la cadena* u a las siguientes expresiones:

$$posicion\,(u)$$

$$\varphi\,(u)$$

6.15.4 Algoritmo para el cálculo del estado de la tortuga asociado a una cadena con interpretación gráfica de tipo tortuga

Definimos recursivamente la función estado(cadena, estado_actual, pila_estados) así:

Sea (D, M, U, α, m) la interpretación gráfica de tipo tortuga.

- estado(λ, estado_actual, pila_estados) = estado_actual
- estado($+.u$, estado_actual, pila_estados) = estado(u, (estado_actual.posicion, estado_actual.inclinación + α)) $\forall u \in D \cup M \cup U$
- estado($-.u$, estado_actual, pila_estados) = estado(u, (estado_actual.posicion, estado_actual.inclinación - α))
- estado($(.u$, estado_actual, pila_estados) = estado(u, estado_actual, push(estado_actual, pila_estados))
- estado($).u$, estado_actual, pila_estados) = estado(u, pop(pila_estados), pila_estados)
- estado($s.u$, estado_actual, pila_estados) = estado(u, ((estado_actual.posicion.x + m $\star$ cos(estado_actual.inclinación), estado_actual.posicion.x + m $\star$ sen(estado_actual.inclinación)), inclinación), pila_estados) $\forall s \in D$
- estado($s.u$, estado_actual, pila_estados) = estado(u, ((estado_actual.posicion.x + $m \star$ cos(estado_actual.inclinación), estado_actual.posicion.x + $m \star$ sen(estado_actual.inclinación)), inclinación), pila_estados) $\forall s \in M$
- estado($s.u$, estado_actual, pila_estados) = estado(u, estado_actual, pila_estados) $\forall s \in U$.

Es estado de la tortuga asociado a una cadena u puede calcularse así:

$$\text{estado}(u,(\ (x_0,y_0),\ \alpha_0),\ [])$$

Donde, [] representa una pila de estados vacía.

6.15.5 Observación

Merece la pena resaltar explícitamente el hecho de que para calcular la inclinación de la tortuga asociada a una cadena con interpretación gráfica de tipo tortuga es suficiente contar el número de veces que aparece el símbolo '+' en la cadena, contar el número de veces que aparece el símbolo '-' en la cadena y multiplicar por α la diferencia entre esas cantidades.

Formalmente:

$$\varphi(x) = \alpha_0 + (\text{ocurrencias('+',x)} - \text{ocurrencias('-',x)})\alpha$$

6.15.6 Definición

Se dice que una *cadena es invariante respecto al ángulo* si, desde el punto de vista de su interpretación gráfica de tipo tortuga cumple las siguientes condiciones:

- No termina dentro de una ramificación.
- La inclinación de la tortuga al comienzo y al final de la interpretación de la cadena permanece inalterada.

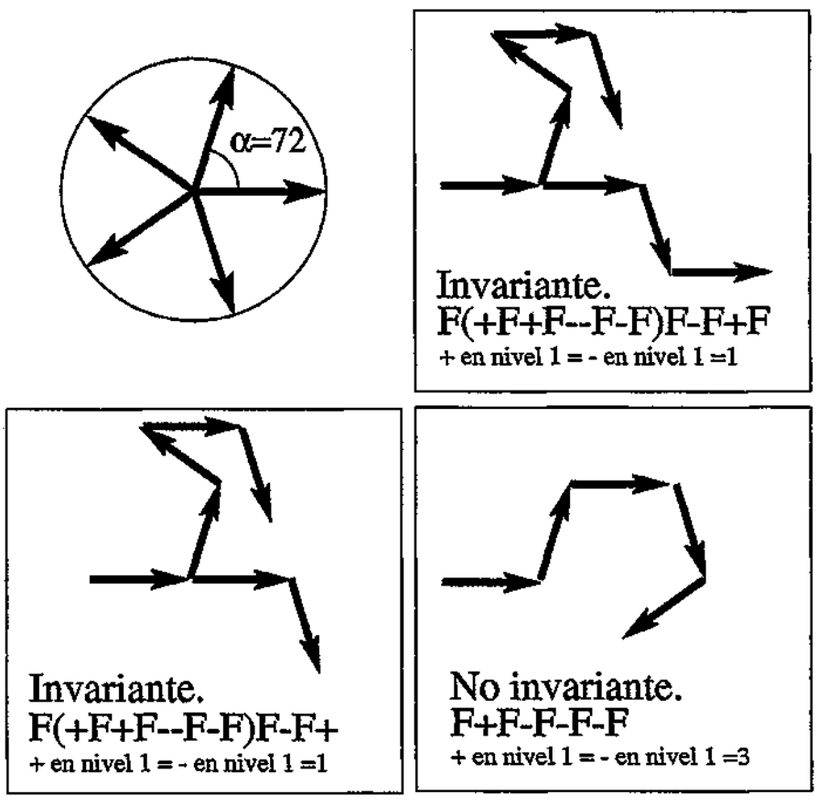


Figura 5.c.a: Las cadenas superior derecha e inferior izquierda son invariantes al ángulo. Obsérvese que la propiedad no depende del último trazo, sino de la inclinación de la tortuga. La figura inferior derecha muestra una cadena no invariante.

6.15.7 Definición

Dado un alfabeto Σ, se define la función *primer_nivel* de la siguiente forma:

$$\text{primer\_nivel}:\Sigma^* \rightarrow \Sigma^*$$

A cada cadena le asigna el primer nivel de ramificaciones de la misma, es decir, lo que queda al quitar de la cadena considerada todos los paréntesis balanceados y lo que hay en su interior.

Ejemplo

Sea $\Sigma = \{0, 1, (,), +, -\}$

Sea $x = 0+1-(0+1-(1+1+1)+0)-(1+1+1)$

$\text{primer\_nivel}(x) = 0+1--$

Ejemplo

Sea $\Sigma = \{F, f, G, (,), +, -\}$

Sea $x = Ff(+Ff - Gf) + G - fG$

primer_nivel$(x) = Ff + G - fG$

6.15.8 Definición, función apariciones

Dado un alfabeto Σ. se define la función *apariciones* de la siguiente forma:

$$apariciones : \Sigma \times \Sigma^* \rightarrow \aleph$$

A cada par (símbolo, cadena) le asigna el número de veces que el símbolo aparece en la cadena.

Ejemplo

Sea $\Sigma = \{0, 1, (,), +, -\}$

Sea $x = 0 + 1 - (0 + 1 - (1 + 1 + 1) + 0) - (1 + 1 + 1)$

apariciones $(0, x) = 3$

apariciones $(1, x) = 3$

apariciones $(+, x) = 7$

apariciones $((, x) = 3$

apariciones $(), x) = 3$

6.15.9 Teorema de caracterización

Enunciado

(Recuérdese que el ángulo de la interpretación tortuga es $\alpha = \frac{2k\pi}{n}$ con $\frac{k}{n}$ irreducible y $k, n \in Z$)

.

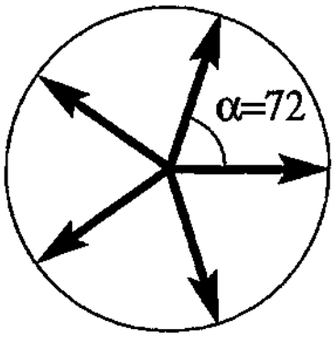
Una cadena es invariante respecto al ángulo si y sólo si los paréntensis que contiene están balanceados (en el sentido habitual en el que se entiende balancear paréntesis en las expresiones en que aparecen) y el número de símbolos '+' menos el número de símbolos '-' en la subcadena no sujeta a paréntesis es 0 o un múltiplo de n.

Justificación informal

Una cadena es invariante al ángulo cuando al interpretarla según los gráficos de tortuga, la cadena no modifica la inclinación inicial de la tortuga. Ante esto puede observarse lo siguiente:

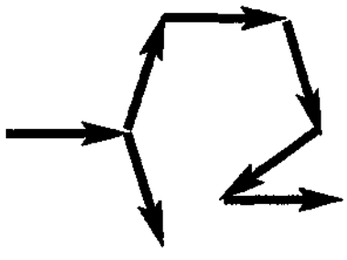
- Las subcadenas que representan ramificaciones no afectan porque estas cadenas ofrecen

siempre ramas completas, es decir, no se puede terminar la cadena sin haber cerrado todas las ramas abiertas. Esto quiere decir que una cadena que contiene ramificaciones es invariante al ángulo de manera independiente a que lo sean las ramas que se abran en ella. La forma de indicar que una rama ha comenzado y terminado es que se encuentren en la cadena tanto su paréntesis de apertura (principio de la cadena) como el de cierre (final de la cadena).



No interesa, rama no cerrada.

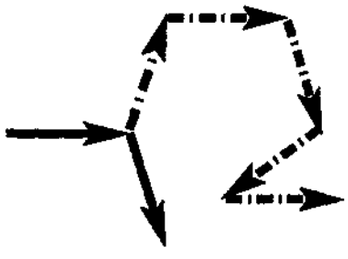
F(+F+F--F-F+F-F



No invariante.

F(+F-F-F-F++F)-F

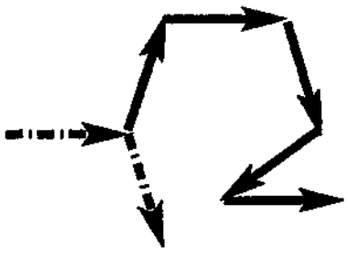
+ en nivel 1=0 y - en nivel 1=1; n=5.



No invariante.

F-F

+ en nivel 1=0; - en nivel 1=1; n=5.



Rama invariante.

+F-F-F-F++F

+ en nivel 1 = 3; - en nivel 1 =3.

Figura 5.c.b: Cadenas invariantes y no invariantes al ángulo.

- Los únicos símbolos capaces de modifcar la inclinación de la tortuga son '+' y '-'. Además son opuestos, en el sentido de que la inclinación de la cadena sumará tantas veces α como símbolos '+' contenga y restará tantas veces α como símbolos '-' contenga.
- Los incrementos de ángulo posible en las interpretaciones tortuga pertenecen al grupo nZ (números enteros módulo n), por tanto todos los números de incrementos múltiplos de n son el 0 de este grupo (elemento neutro para la suma).

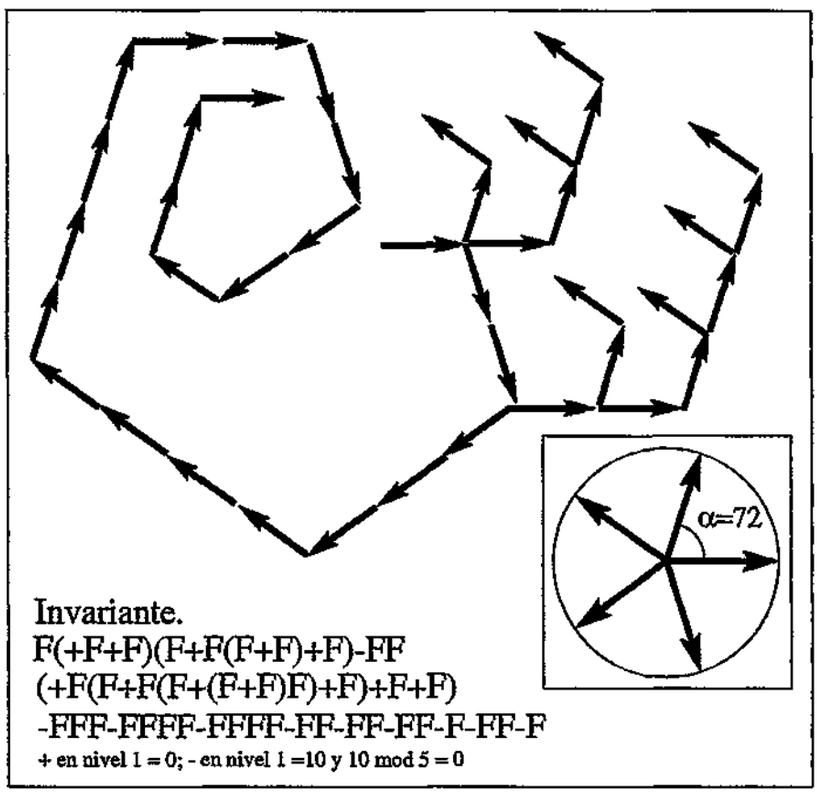


Figura 5.c.c: Ejemplo de cadena invariante al ángulo: obsérvese que en este caso la diferencia entre el número de símbolos '+' y '−' es múltiplo de n(=5) y que hay ramas no invariantes.

6.16 Gráficos tortuga de Lindenmyaer invariantes respecto al ángulo

6.16.1 Definición

Se dice que esquema TGD0L de un gráfico tortuga de Lindenmayer es invariante respecto al ángulo (a partir de ahora AITGD0L) si y sólo si la parte derecha de todas sus reglas son cadenas invariantes respecto al ángulo.

6.16.2 Definición

Se dice que un sistema TGD0L de un gráfico tortuga de Lidenmayer es AITGD0L si y sólo si su esquema lo es.

A partir de ahora se mencionará indistintamente esquema, sistema o gráfico de Lindenmayer invariante al ángulo cuando el contexto elimine las posibles ambigüedades.

6.16.3 Observación

El presente trabajo se centra en este tipo de sistemas porque la mayoría de los sistemas TGD0L que aparecen en la literatura pertenecen al tipo AITGD0L. Es cierto que no todos lo son pero es posible encontrar sistemas AITGD0L gráficamente equivalentes a muchos sistemas TGD0L que no son AITGD0L. Esta última circunstancia se muestra en el siguiente ejemplo.

6.16.4 Ejemplo

Sea el gráfico tortuga de Lindenmayer compuesto por el siguiente sistema TGD0L y la siguiente interpretación gráfica:

$$GTL_1 = \left(\begin{array}{c} LS_1 = \left(\begin{array}{c} \{F, G, +, -\}, \\ P_1 = \left\{ \begin{array}{l} F ::= F + G, \\ G ::= F - G, \\ + ::= +, \\ - ::= - \end{array} \right\}, \\ F \end{array} \right) \\ G_{T_1} = \left(D = \{F, G\}, M = \{\}, U = \{\}, \alpha = \frac{2\pi}{4}, m = 1\right) \end{array} \right)$$

LS_1 no es invariante al ángulo, pues ocurrencias('+',F+G)-ocurrencias('-',F+G) = 1 que no es ni 0 ni múltiplo de 4

La interpretación gráfica de la cadena obtenida cuando el número de derivaciones tiende a infinito es la bien conocida curva fractal del dragón, como se puede apreciar en la gráfica 5.c.d que muestra la décima derivación.

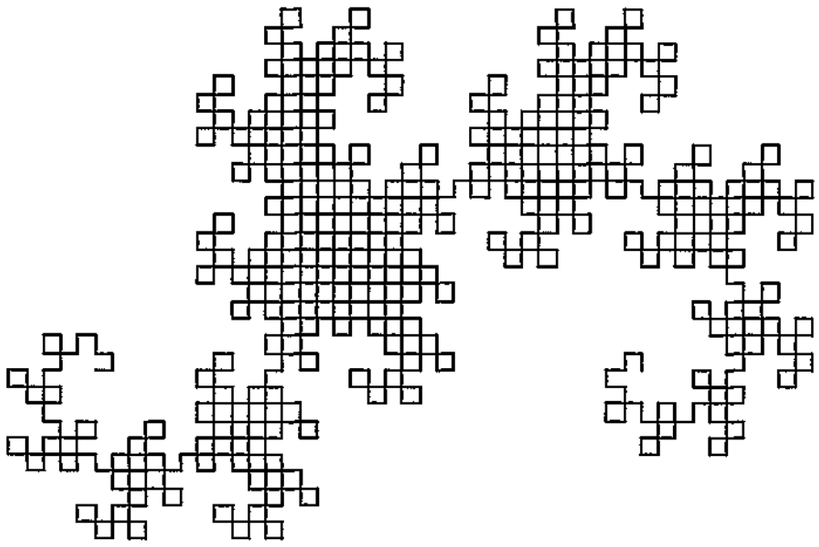


Figura 5.c.d: Décima derivación: el dragón.

A continuación se muestra otro gráfico tortuga de Lindenmayer gráficamente equivalente al anterior. Su sistema es AITGD0L es equivalente al sistema LS_1 con la interpretación gráfica G_{T_2}.

$$GTL_2 = \left(\begin{array}{c} LS_2 = \left(\begin{array}{c} \Sigma = \{F, G, +, -\}, \\ P_2 = \left\{ \begin{array}{l} F ::= F + G -, \\ G ::= +H - G, \\ H ::= H + I -, \\ I ::= +F - I, \\ + ::= +, \\ - ::= - \end{array} \right\}, \\ F \end{array} \right) \\ G_{T_2} = \left(D = \{F, G, H, I\}, M = \{\}, U = \{\}, \alpha = \frac{2\pi}{4}, m = 1 \right) \end{array} \right)$$

Aunque no hayamos podido demostrar que todos los sistemas GTD0L utilizados para representar fractales sean gráficamente equivalentes a algún sistema AIGTD0L, sí hemos conseguido hacerlo en una serie de casos, mostrando al menos la generalidad e interés del resultado.

6.17 Representación n-dimensional de las cadenas con interpretación de tortuga

En el capítulo 8 se analiza la conveniencia de utilizar una representación de los puntos de las interpretaciones gráficas de las cadenas de sistemas L distinta de los pares de $\Re^2$. En las siguientes secciones se define la que será utilizada entonces.

6.17.1 De pares de números reales a vectores n-dimensionales de enteros

Notación.

Los vectores serán nombrados o bien mediante una letra que los identifique (siempre que el contexto deje claro que son vectores), o bien mediante los puntos que definen sus extremos (siempre que el contexto deje claro cuál es el origen y cuál el punto final).

El módulo de un vector (v) seguirá la notación habitual ($|v|$).

El ángulo de un vector (v) respecto al eje positivo de las x se representará como $\varphi(v)$

Lema

Recuérdese que en un sistema AITGD0L con una interpretación gráfica de tortuga G_T de ángulo $\alpha = \frac{2k\pi}{n}$, todos los movimientos elementales de la tortuga pueden representarse mediante n vectores unitarios de ángulos $i\alpha$,, $i \in nZ = \{0, ..., n-1\}$ y que se ha decidido en las secciones precedentes llamar $V(G_T)$ al conjunto de esos vectores.

En estas condiciones resulta trivial comprobar que se cumple que

$$V(G_T) \text{ contiene vectores opuestos} \Leftrightarrow n \text{ es par.}$$

Definición

En la situación descrita en el lema anterior, se define como *camino tortuga* C un conjunto de vectores encadenados pertenecientes a $V(G_T)$:

$$C = \left\{\left(p_i^o, p_i^d\right)\right\}_{i\in\{0,..,m\}} | p_i^d = p_{i+1}^o \forall i$$

Donde

- Cada vector está siendo representado por un par de puntos que son sus extremos y que han sido descritos utilizando la siguiente notación:
  - p_i^o es el punto *o*rigen del vector i-ésimo.
  - p_i^d es el punto *d*estino del vector i-ésimo.

La figura 6.c.1 muestra un camino para con una G_T con $k = 1$ y $n = 8$ ($\alpha = 45^0$).

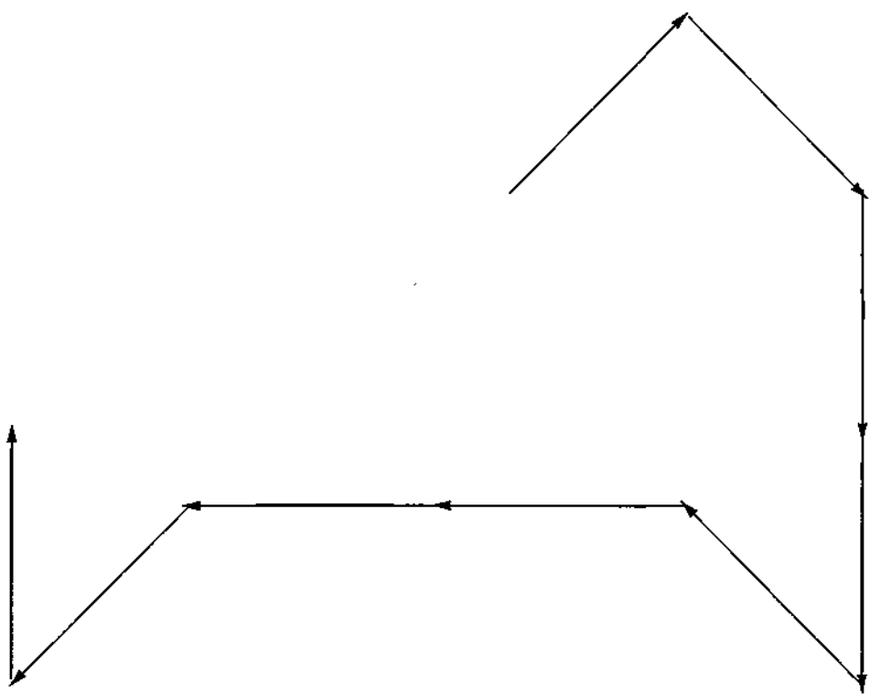


Figura 6.c.1: Camino tortuga.

Definición

Se define *origen de un camino tortuga* C, y se representa *origen* (C) como el punto origen de su primer vector.

Sea $C = \left\{\left(p_i^o, p_i^d\right)\right\}_{i\in\{0,..,m\}}$ un camino, *origen* $(C) = p_0^o$.

Definición

Se define como *destino de un camino tortuga* C, y se representa *destino* (C), al punto destino de su último vector.

Sea $C = \left\{\left(p_i^o, p_i^d\right)\right\}_{i\in\{0,..,m\}}$ un camino, *destino* $(C) = p_m^d$.

Definición

Se define como *longitud de un camino tortuga* C, y se representa $|C|$, a su cardinal.

$$|C| = \#(C)$$

Definición

Se define como *ciclo tortuga* todo camino tortuga C en el que se cumple la condición de que los dos únicos elementos que son iguales son el primero y el último.

C ciclo $\Leftrightarrow$

- C es camino y
- $C = \{(p_i^o, p_i^d)\}_{i\in\{0,..,m\}}\ (p_0^o, p_0^d) = (p_m^o, p_m^d)$ y $(p_k^o, p_k^d) \neq (p_l^o, p_l^d)\ \forall k \neq l \neq 0 \neq m$

La figura 6.c.2 muestra un ciclo tortuga.

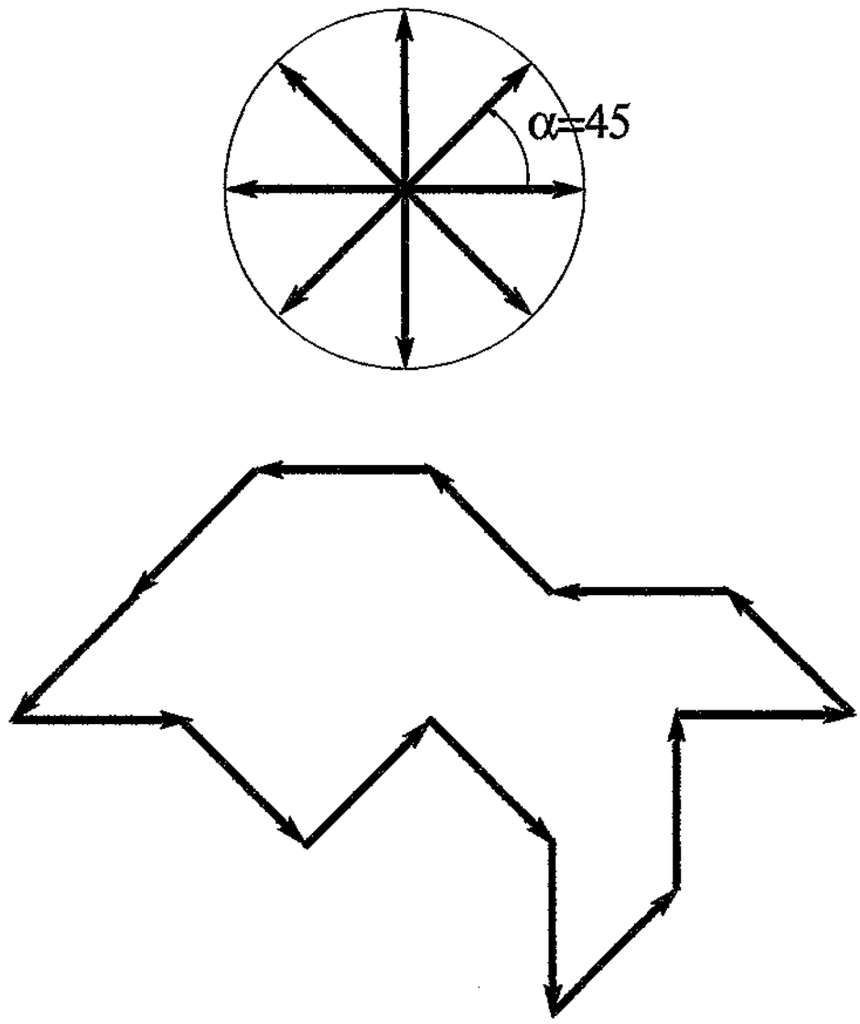


Figura 6.c.2: Ciclo tortuga.

Notación para los caminos tortuga.

Se puede utilizar otra notación para los caminos tortuga debido a sus siguientes propiedades:

- Todos los vectores comparten el mismo módulo, que puede tomarse como unidad.
- A partir de las características de los ángulos de los vectores, y ya que están encadenados, en lugar de representarlos mediante la mención explícita de sus extremos puede utilizarse el número de veces que hay que incrementar el ángulo elemental del camino para obtener el ángulo de cada vector.

De esta forma una notación alternativa para un camino tortuga es una tupla con las siguientes componentes:

- El módulo que comparten todos los vectores .
- El punto de inicio del camino.
- El ángulo mínimo.
- Los incrementos de ese ángulo asociados a los correspondientes ángulos de cada vector del camino.

El camino de la figura 6.c.1 podría representarse de la siguiente manera:

$$(1, (0,0), \tfrac{\pi}{4}, (1,7,6,6,3,4,4,5,2))$$

Observación: cálculo del origen de un camino tortuga

Sea $T = (m, (x_o, y_o), \alpha, (k_1, ..., k_p))$ un camino tortuga , $origen\,(T) = (x_o, y_o)$

Observación: cálculo del destino de un camino tortuga

Las coordenadas cartesianas del destino de un camino tortuga T , pueden ser calculadas a partir de los ángulos de los vectores y del módulo.

Sea $T = (m, (x_o, y_o), \alpha, (k_1, ..., k_p))$ un camino tortuga , $destino\,(T) = (\, x_o + m \sum_{i=1}^{p} \cos(k_i \alpha)$, $y_o + m \sum_{i=1}^{p} \sin(k_i \alpha)\,)$

Observación: cálculo de la longitud de un camino tortuga

La longitud de un camino tortuga T puede calcularse como el cardinal de su componente de incrementos de ángulo.

Sea $T = (m, (x_o, y_o), \alpha, (k_1, ..., k_l))$ un camino tortuga , $|T| = |(k_1, ..., k_l)| = l$

Lema, invariancia de la longitud del camino tortuga y del destino respecto a la posición de los segmentos

La longitud y el destino de un camino tortuga no depende del orden en el que se encadenan sus vectores.

Sea $T = (m, (x_o, y_o), \alpha, d)$ camino tortuga $\Rightarrow$

$\forall T' = (m, (x_o, y_o), \alpha, d'), , d' \in permutaciones(d)$[1]

1. $destino(T) = destino(T')$
2. $longitud(T) = longitud(T')$

Demostración

Se deduce directamente de la propiedad conmutativa de la suma de vectores.

Ejemplo.

La figura 6.c.3 muestra un ejemplo en el que se comprueba que el camino cuya representación es $(1, (0,0), \frac{\pi}{4}, (1,2,3,4,4,5,6,6,7))$ tiene el mismo destino y la misma longitud que el de los ejemplos anteriores.

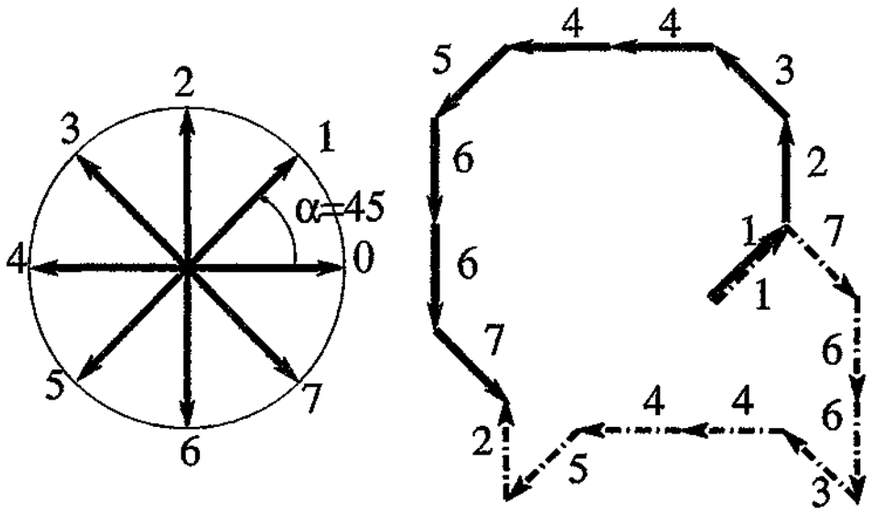


Figura 6.c.3: Destino y longitud invariantes al orden.

[1] *permutaciones(d)* es el conjunto formado por todas las tuplas que tienen los mismos elementos que d pero en diferente orden.

Definición

Puesto que la posición que ocupen los vectores en la cadena es indiferente, la componente de incrementos de α de la representación se mostrará desde ahora en orden creciente tal y como se ha hecho en el ejemplo anterior. A esta notación se la llamará *notación ordenada de un camino L tortuga.*

Observación, origen (0,0) de los caminos L tortuga considerados

En los caminos L tortuga que aparecerán a lo largo de este trabajo se supondrá que el origen es siempre el punto (0,0) y el módulo de los vectores 1. Por lo tanto un camino tortuga T tal que $destino\,(T) = (x, y)$ y $origen\,(T) = (0, 0)$ podrá ser utilizado en lugar de las coordenadas cartesianas para expresar el punto (x, y) de $\Re^2$.

Definición

Se extenderá la definición de conjunto de ángulos de una interpretación gráfica a un camino. De esta forma

- $\Theta\,(c)$ representará el *conjunto de ángulos de los vectores del camino c.*

Formalmente

$$\Theta\,(c)=\left\{\arctan\left(\frac{y}{x}\right) \quad \forall\,(x,y) \in c\right\}$$

Definición

Se llamará *conjunto de incrementos de un camino tortuga c* y se escribirá $\Theta_n\,(c)$ al conjunto de los múltiplos del ángulo α de la interpretación gráfica utilizada asociados a los elementos de $\Theta\,(c)$.

Formalmente

$$\Theta_n\,(c)=\{p \in Z| \quad \exists\psi \in \Theta\,(c)\,,\,\psi = p\alpha\}$$

Es, por tanto, posible adoptar una representación más compacta, pues sólo se necesita conocer el ángulo de la interpretación gráfica, y el número de vectores de cada tipo que aparecen en el camino.

A lo largo de este trabajo se ordenará las componentes asociadas a cada vector por el número de incrementos del ángulo α asociado a cada uno de ellos.

Es necesario, a la hora de definir una representación compacta útil para estos caminos, comprobar si los tipos de símbolos de las cadenas que serán estudiadas obligan a guardar más información.

Como se mostró en la primera sección de este capítulo, se permite utilizar símbolos de los siguientes tipos:

- Dibujables, se desplaza la cabeza de la tortuga dejando un rastro visible.
- De movimiento, se desplaza la cabeza de la tortuga sin dejar rastro visible.
- De incremento de ángulos (+ y -).
- No gráficos pero necesarios para las derivaciones, que simplemente se ignoran.
- De ramificación (comienzo de rama y fin de rama).

El tratamiento de todos estos símbolos, excepto los de movimiento, podría realizarse con las notaciones presentadas hasta ahora. Estos tienen que ser tratados geométricamente (cálculo de longitud, destino, etc...) como si fueran dibujables, porque realmente se desplaza la tortuga, pero de forma invisible.

Para indicar si una representación compacta de un camino realiza un movimiento invisible se necesita una nueva componente: la visibilidad del camino.

Definición, representación compacta L tortuga para una cadena de una gráfico tortuga de Lindenmayer

Se define como *representación compacta L tortuga de una cadena* (que llamaremos *palabra*) *de un gráfico tortuga de Lindenmayer* $((S, G_T))$, y se escribirá G_T (*palabra*), a una quíntupla

- $P_t\left(m,(0,0),\frac{2k\pi}{n},(\alpha_0,...,\alpha_i,...\alpha_{n-1}),v\right)$, donde todas las componentes (excepto v) se han definido anteriormente en otras representaciones de los caminos excepto la última: v, que lleva cuenta de la característica de visibilidad del punto representado tomando el valor 1 para indicar que es visible y 0 para indicar que es invisible. Por ejemplo si se supone que los caminos mostrados en los ejemplos anteriores terminan en puntos visibles, la representación compacta para cualquiera de ellos es la siguiente:

$$\left(1,(0,0),\tfrac{\pi}{4},(0,1,1,1,2,1,2,1),1\right)$$

Obsérvese que las dos primeras componentes pueden considerarse redundantes si se considera, como es frecuente, que el módulo es 1 y el origen $(0,0)$. Aunque esto es así en todos los casos estudiados en este trabajo se mantiene todas la representación completa para no perder generalidad.

Esta representación es obtenida de la siguiente manera a partir de la cadena *palabra*:

Algoritmo para la obtención de la representación compacta L tortuga para una cadena de un gráfico tortuga de Lindenmayer

Llamemos a este algoritmo

palabra_a_representación_compacta(palabra, representación, posición_ángulo,posición_palabra,)

Donde

- *palabra* es la cadena que se está estudiando. Es una de las que hace al punto estudiado accesible por el gráfico..
- *representación* es la representación actual. Se construye de forma recursiva.
- *posición_ángulo* lleva cuenta del valor de índice que sufrió la última modificación en la componente de incrementos de ángulo de la representación compacta.
- *posición_palabra* lleva cuenta del siguiente valor de índice sobre la palabra que tiene que ser estudiado.

Y devuelve

- La representación del punto extremo de la cadena.
- La última posición modificada en la componente de incrementos de ángulo de la representación.

El algoritmo se muestra a continuación, se utilizará la notación ¡vector¿[¡indice¿] para acceder a la valor que ocupa la posición indicada por ¡indice¿ en el vector ¡vector¿, se utilizará también las funciones *dibujable(¡símbolo¿), movimiento(¡símbolo¿) y vacía(¡palabra¿, ¡posición_palabra¿)* para comprobar si un símbolo es dibujable, de movimiento y si a partir de una posición determinada sobre una palabra concreta ya no hay más símbolos:

1. **SI** *vacía(palabra, posición_palabra)* **ENTONCES** *salir*, devolviendo la representación y la posición pasadas como argumentos.
2. **SI** *dibujable(palabra[posición_palabra])*
   - **ENTONCES**
     - *representación[3][posición_ángulo]* ← *(representación[3][posición_ángulo]) +1* (incrementos de segmentos con esa inclinación)
     - *representación[4]* ← *1* (si la cadena termina en este punto, es visible).
3. **SI** *movimiento(palabra[posición_palabra])*
   - **ENTONCES**
     - *representación[3][posicion_ángulo]* ← *(representación[3][posicion_ángulo]) + 1* (incrementos de segmentos con esa inclinación)
     - *representación[4]* ← *0* (si la cadena termina en este punto, es visible).
4. **SI** palabra[posición_palabra] = ')' (final de una rama, esto constituye un caso terminal porque la interpretación de las ramas se concatena)
   - **ENTONCES** *salir*, devolviendo la representación y la posición pasadas como argumentos.

5. **SI** *palabra[posición_palabra]* = '+' **ENTONCES** *posicion_ángulo* ← *(posición_ángulo + 1)* $_{mod(n)}$
6. **SI** *palabra[posición_palabra]* = '-' **ENTONCES** *posicion_ángulo* ← *(posición_ángulo - 1)* $_{mod(n)}$
7. **SI** *palabra[posición_palabra]* = '(' (comienzo de una rama)
   - **ENTONCES**
     - *posición_palabra* ← *posición_palabra + 1*
     - *posición_ángulo_rama* ← *posición_ángulo* (para preservar el caracter "local" de la rama)
     - *palabra_a_representación_compacta(palabra, representación, posición_ángulo_rama, posición_palabra)* (sin embargo, aunque la posición de ángula modificada por última vez se preserva, para continuar con la que se tenía antes de iniciar la rama, es obligatorio modificar de forma "global" la posición sobre la palabra para no volver a analizar la rama y entrar en bucles infinitos)
8. (Común a todas las ramas anteriores que no hayan salido de forma directa con la sentencia salir)
   - *posición_palabra* ← *posición_palabra + 1*
   - *palabra_a_representación_compacta(palabra, representación, posición_ángulo, posición_palabra)*

Definición

Se dice de un *elemento p de $\Re^2$ que es accesible mediante un gráfico tortuga de Lindenmayer G si* y sólo si existe al menos una palabra del lenguaje de G cuya interpretación gráfica sea un camino tortuga que lo tenga como destino.

Formalmente:

- Sea $p \in \Re^2$.
- Sea $G = (S, G_T)$ un gráfico tortuga de Lindenmayer

$$p \text{ accesible tortuga} \Leftrightarrow \exists x \in L(S) \wedge destino\left(G_T\left(x\right)\right) = p$$

Cálculo del destino de un camino L tortuga a partir de su representación compacta.

El cálculo del destino tiene que reflejar también que ahora los segmentos de la misma longitud están agrupados.

Sea $T_c = \left(m, (x_o, y_0), \frac{2k\pi}{n}, (\alpha_0, ..., \alpha_i, ...\alpha_{n-1}), v\right)$,

$$destino(T_c) = \left(x_o + m\sum_{i=0}^{n-1} \alpha_i \cos\left(i\alpha\right), y_0 + m\sum_{i=0}^{n-1} \alpha_i \sin\left(i\alpha\right)\right)$$

Donde α_i es el número de vectores de ángulo $i\alpha$ que forman parte del camino.

Cálculo de la longitud de un camino L tortuga a partir de su representación compacta

La longitud de un camino L tortuga representado de forma compacta es la suma de las componentes de su tupla de incrementos de ángulo.

Sea $T_c = \left(m, (x_0, y_0), \frac{2k\pi}{n}, (\alpha_0, ..., \alpha_i, ...\alpha_{n-1}), v\right)$,

$$|T_c| = \sum_{i=0}^{n-1} \alpha_i$$

Donde α_i es el número de vectores de ángulo $i\alpha$ que forman parte del camino.

Definición, representación compacta L tortuga para un punto del plano $\Re^2$ accesible por un gráfico tortuga de Lindenmayer

Se define como *representación compacta L tortuga de un punto del plano $\Re^2$ accesible por un gráfico de Lindenmayer* a la representación compacta de cualquiera de las cadenas que hacen que el punto sea accesible.

Teorema, caracterización de ciclos L tortuga

Un camino L tortuga es cíclico si y sólo si el conjunto formado por los incrementos de ángulo de sus vectores $(\Theta_n(c))$ contiene algún grupo cíclico finito.

Formalmente

$$c \text{ camino tortuga cíclico} \Leftrightarrow \exists m \in Z, \Psi \subseteq \Theta_n(c) \mid \Psi \text{ y } Z/mZ \text{ son isomorfos.}$$

Justificación

Los resultados de la teoría de grafos afirman que todo grupo cíclico (monógeno) A cumple

- Si A es infinito $\Rightarrow$ A es isomorfo a Z .
- Si A es finito $\Rightarrow \exists m \in Z \mid A$ y Z/mZ son isomorfos.

Donde $Z/mZ = \{[0], [1], ..., [m-1]\}$ y cada una de las clases de equivalencia $[i] = \{ki \quad \forall k \in Z\}$, es decir son los múltiplos de i.

El enunciado tiene dos partes:

Directa Un camino L tortuga cíclico c cumple que $\Theta_n(c)$ contiene un grupo cíclico finito.

Inversa Todo camino L tortuga c cuyo $\Theta_n(c)$ contiene un grupo cíclico finito es un camino L tortuga cíclico.

El cumplimiento de las dos partes corresponde a propiedades geométricas y algebraicas elementales. A continuación se mostrará algunas figuras y se realizará algunas observaciones que, aunque no suplen la demostración, sí revelan la trivialidad del enunciado.

Los ciclos contienen en sus Θ_n grupos cíclicos finitos

La figura 6.c.5 muestra algunos grupos cíclicos contenidos en los $\Theta_n(c)$ de algunos caminos tortuga en una interpretación gráfica con $n = 16$. En la parte superior se muestra los vectores permitidos por la interpretación. El resto de las figuras muestra caminos que incluyen ciclos. Obsérvese que todos los ciclos contienen parejas de vectores opuestos. De hecho al ser $n = 16$ (par) el menor ciclo posible es de longitud 2 y corresponde a una pareja de vectores opuestos. Todas las parejas de vectores opuestos son isomorfas a $Z/2Z = \{[0], [1]\}$

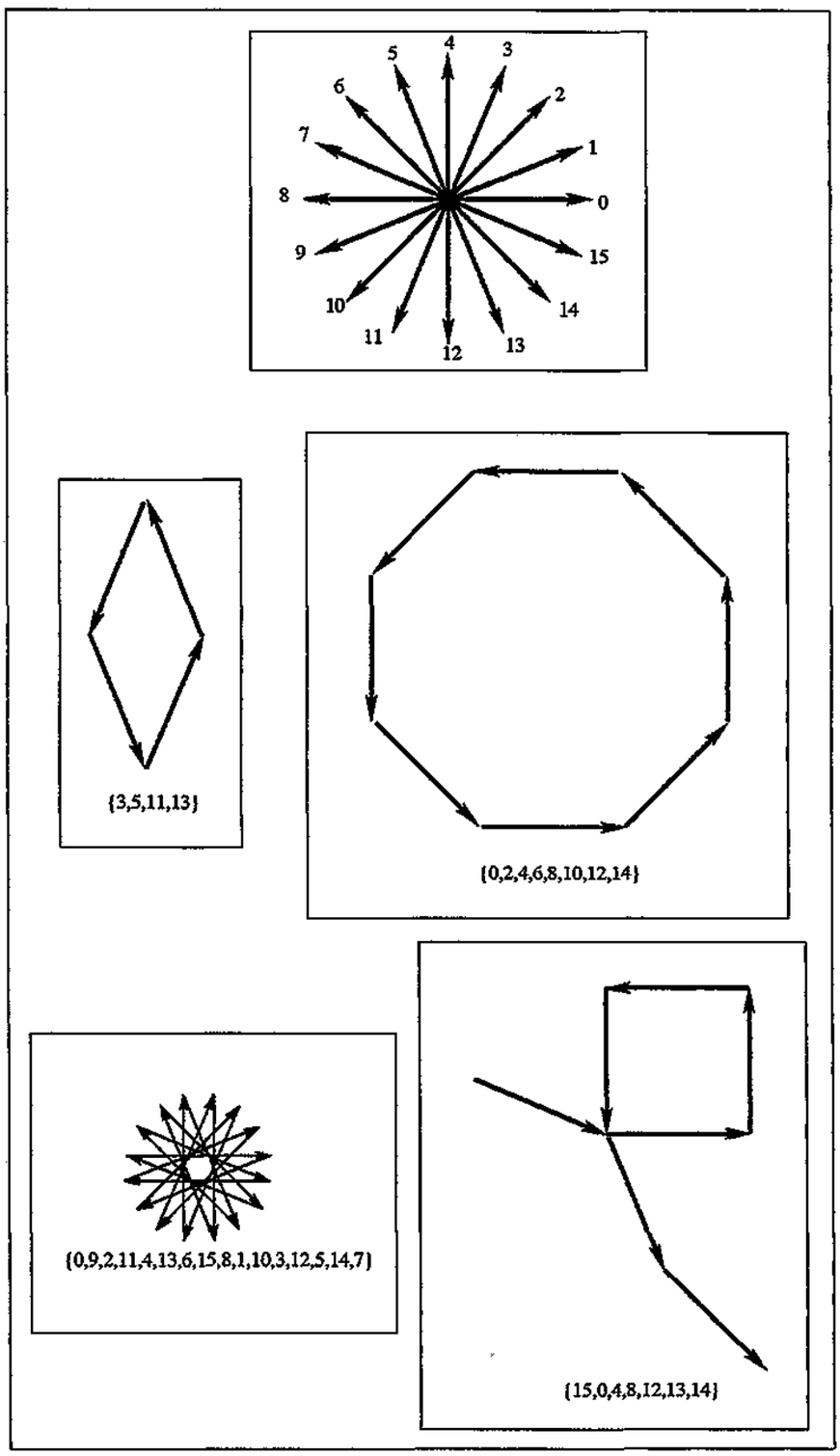

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
{3,5,11,13}
{0,2,4,6,8,10,12,14}
{0,9,2,11,4,13,6,15,8,1,10,3,12,5,14,7}
{15,0,4,8,12,13,14}

Figura 6.c.5: Identificación de ciclos $n = 16$.

La figura 6.c.6 hace algo similar con una interpretación gráfica con $n = 15$.

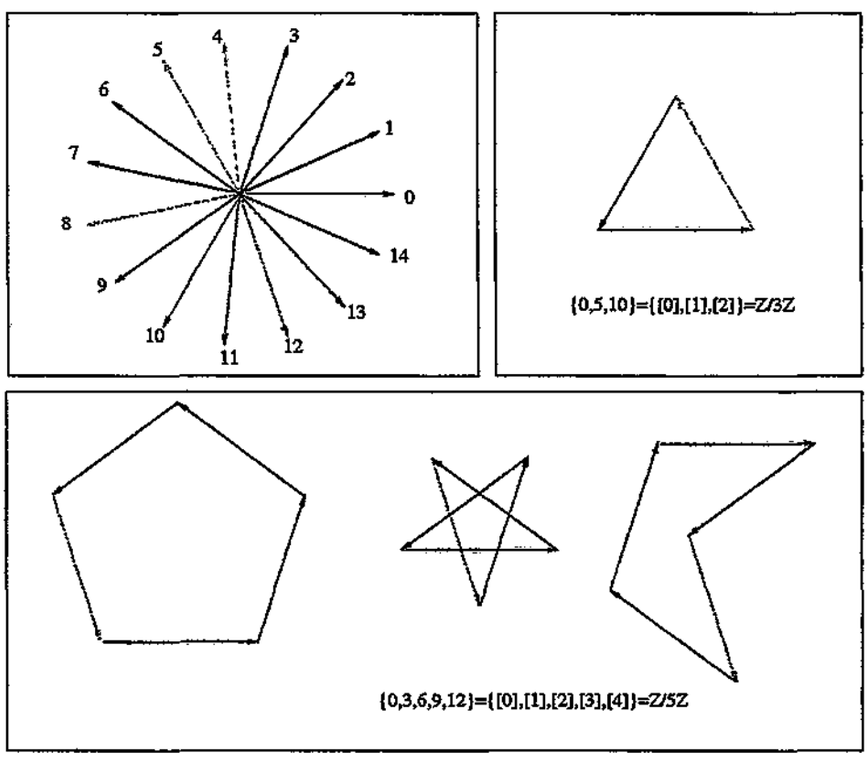


Figura 6.c.6: Identificación de ciclos $n = 15$.

En la parte superior izquierda se muestra los vectores permitidos por la interpretación gráfica.

En la parte superior derecha se muestra el ciclo de longitud 3 c_1. El texto indica el isomorfimo entre $\Phi_n(c_1)$ y $Z/3Z$.

En la parte inferior se muestra tres ciclos (c_2, c_3, c_4) con el mismo conjunto de incrementos de α $(\Phi_n(c_2) = \Phi_n(c_3) = \Phi_n(c_4))$. El texto indica el isomorfismo entre los tres conjuntos y $Z/5Z$.

Los grupos cíclicos finitos utilizados como Φ_n generan ciclos

La figura 6.c.7 muestra un ejemplo en el que se aprecia las razones geométricas que hacen a los caminos c cuyos $\Phi_n(c)$ contienen grupos cíclicos finitos ser caminos cíclicos. Se muestra el camino cíclico conseguido con los 9 vectores permitidos con una interpretación gráfica tortuga con $n = 9$. Los incrementos de ángulo del camino formado por esos vectores forman el grupo cíclico Z_8 que es isomorfo a $Z/9Z = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$. La reflexión puede generalizarse a cualquier valor de n. La figura geométrica que consta de n segmentos de longitud fija encadenados de manera

que la inclinación de cada uno de ellos difiere con la del anterior en $\alpha = \frac{2k\pi}{n}$ es un polígono regular de n lados. Es evidente que el sentido del recorrido inducido por la dirección de los vectores del camino es un ciclo de longitud n. En la figura las flechas unen ángulos cuya suma es π.

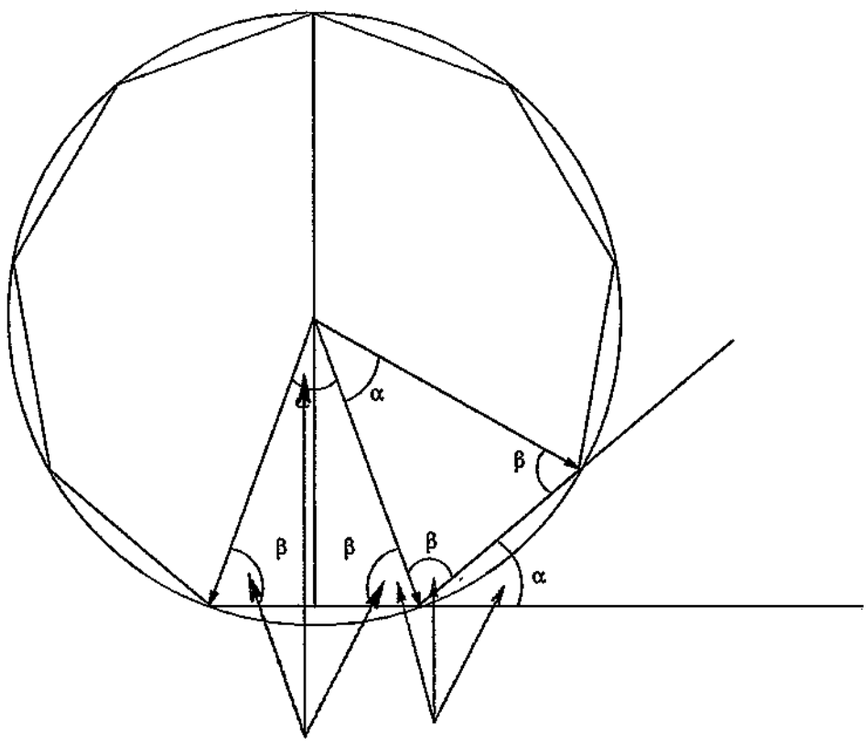


Figura 6.c.7: Z/nZ es un ciclo tortuga con $n = 9$.

Por tanto, en los polígonos regulares el mismo ángulo que separa los radios de la circunferencia circunscrita que unen vértices contiguos (α) es la diferencia en ángulo que separa las rectas sobre las que se encuentran dos lados contiguos cualesquiera.

Es decir, el recorrido tortuga que consiste en realizar un trazo de una longitud con una inclinación inicial γ, incrementar esa inclinación en en ángulo $\alpha = \frac{2k\pi}{n}$ y repetir el proceso n-1 veces genera el polígono regular de n lados.

Resultados de teoría de grupos. A continuación se recuerda algunos enunciados de teoría de grupos

- Si p es un número primo y A es un grupo de orden (cardinal) p, G es cíclico.
- Sea A un grupo cíclico de orden n_A. Para cada divisor m de n_A existe un único subgrupo de A de orden m. Además este subgrupo es cíclico.
- Todo subgrupo de un grupo cíclico es cíclico.

Las demostraciones de estos resultados están ampliamente publicadas en la literatura de la materia y pueden ser consultadas en la bibliografía.

Puesto que en el resultado de caracterización de los ciclos tortuga se ha hecho una correspondencia entre caminos cíclicos de p tramos y grupos cíclicos de p elementos es claro el significado gráfico que tienen estos resultados.

Teorema

Los ciclos tortuga pueden ser recorridos en los dos sentidos posibles si y sólo si n es par (n es la constante de la interpretación gráfica tortuga tal que $\alpha = \frac{2k\pi}{n}$.

Justificación

Este teorema se deduce directamente de la proposición relativa a la existencia de vectores opuestos en una interpretación gráfica si y sólo si el valor n de la interpretación gráfica es par. De la paridad de n se deduce que puede generarse el camino opuesto a cualquiera otro. Eso también es cierto para los ciclos.

En la figura 6.c.8 se muestra los recorridos en los dos posibles sentidos de un ciclo.

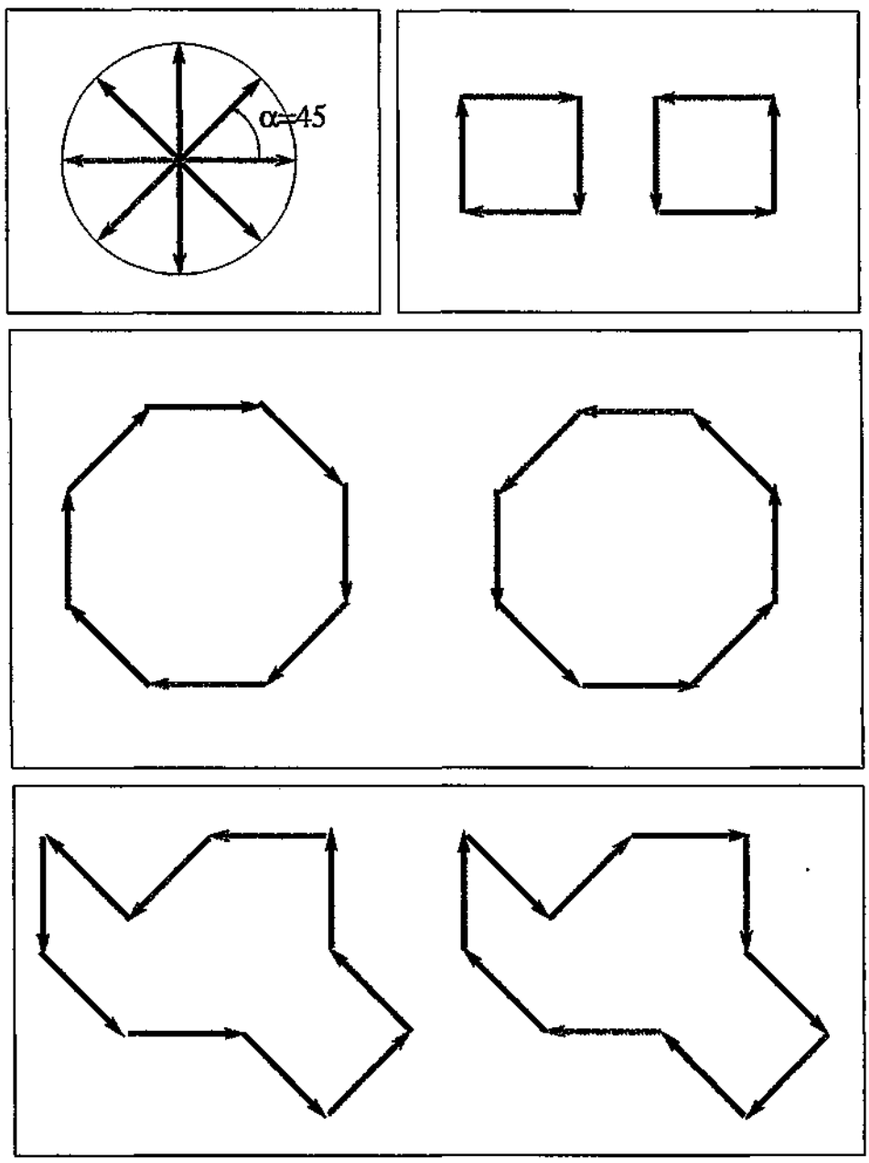


Figura 6.c.8: Ciclos bidireccionales con $n = 8$.

Corolario

Como consecuencia del anterior teorema si n es impar los ciclos tortuga pueden ser recorridos sólo en uno de los dos sentidos posibles.

Lema

El menor conjunto de vectores que incluye a $V(G_T)$, para una G_T con n impar, y a todos sus opuestos es $V\left(G'_T\right)$ (con G'_T obtenida a partir de G_T pero utilizando en lugar de n, $n' = 2n$).

Justificación intuitiva

Este resultado se deduce también fácilmente de las proposiciones anteriores relativas a la paridad de n necesaria para vectores opuestos y a la naturaleza cíclica de los ángulos permitidos en los vectores de las interpretaciones gráficas de tortuga.

Teorema, representación canónica

Dado un gráfico tortuga de Lindenmayer G todo elemento p de $\Re^2$accesible por G puede ser expresado de forma única por una representación compacta de longitud mínima desde el origen hasta el elemento p. A esta representación se la llamará *representación canónica de p* y se escribirá $canon(G,p)$ o simplemente $canon(p)$ cuando no haya confusión posible respecto al gráfico de Lindenmayer considerado.

Justificación intuitiva

Se demostrará este resultado de forma constructiva proponiendo un algoritmo que construye la representación canónica definida a partir de una de las cadenas cuya interpretación gráfica llega a p.

La representación construida por el algoritmo tiene que cumplir las siguientes condiciones:

- Ser única. Ello se consigue porque la representación se hace invariante al orden de los vectores del camino.
- Ser de longitud mínima. Esta condición implica las siguentes:
  - Eliminar ciclos. El algoritmo eliminará de la representación todos los ciclos asegurándose de que no quedan grupos cíclicos finitos en sus componentes de descripción de ángulos.
  - Obtener siempre el camino tortuga más corto entre dos puntos. Las interpretaciones gráficas de tipo tortuga que permiten encontrar varios caminos entre dos puntos son aquellas que tiene un n par. Los puntos de las curvas son siempre vértices de polígonos con ángulos permitidos por la interpretación gráfica. Se ha demostrado previamente que sólo las interpretaciones con n par permiten recorrer los ciclos en los dos sentidos posibles.
    * En las figuras 6.c.9 y 6.c.10 se muestra la interpretación gráfica de cadenas de este tipo en las que se puede elegir entre ir por una parte de un ciclo (en un sentido) o por la otra (recorriendo el ciclo más pequeño al que pertenecen los dos puntos tratados en el sentido contrario). En general un lado del ciclo tendrá una

longitud menor o igual que el otro por lo que para garantizar que la representación canónica elige el camino más corto es necesario llevar cuenta de esta circunstancia.

* Si un lado del ciclo tiene la misma longitud que el otro, los dos caminos posibles coinciden cuando se hacen invariantes al orden de los vectores.
* Esta situación no puede producirse con interpretaciones gráficas con n impar. En ellas si se tiene que un camino une dos vértices de un polígono regular éste sólo podrá recorrerse en el sentido de la cadena y no habrá una posible alternativa de menor longitud.
* Las dos figuras utilizan una interpretación gráfica tortuga con $n = 8$. En ambas se muestra dos puntos llamados origen y destino. Se supone que la cadena tomada como punto de partida por el algoritmo pasa por esos dos puntos y es importante elegir el camino entre ellos de longitud mínima. Las gráficas muestran el mejor camino con trazo discontinuo.

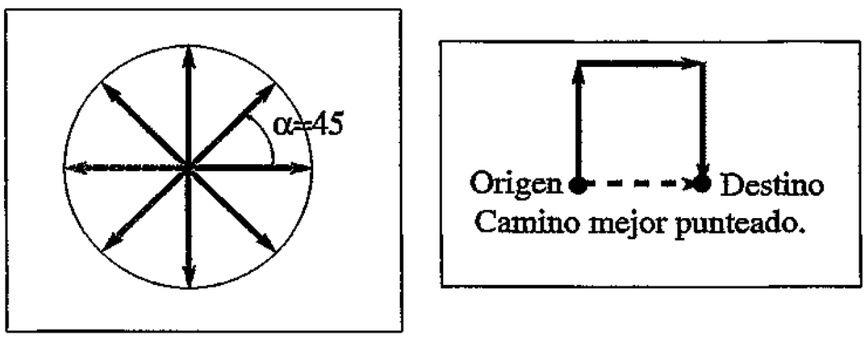


Figura 6.c.9: Ejemplo de elección de camino más corto con $n = 8$.

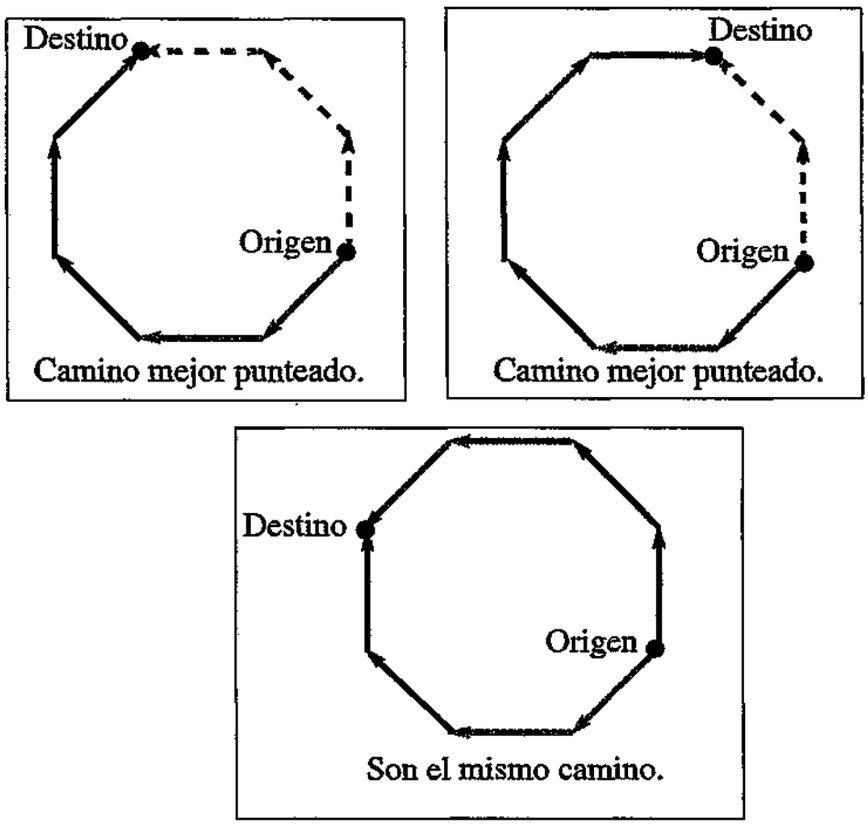


Figura 6.c.10: Ejemplo de elección de camino más corto con $n = 8$.

Demostración

Algoritmo para la obtención de la representación canónica L tortuga para un punto del plano $\Re^2$ accesible mediante un gráfico tortuga de Lindenmayer.

Se calcula la representación compacta de la cadena.

Se elimina los ciclos de longitud n (para ello se resta a todos los elementos de la representación el mínimo) hasta que se obtenga algún cero.

- Si n es par.
  - Se elimina las parejas de vectores opuestos: para ello se reescribe la cadena en forma de matriz $2 \times N/2$ y se resta por columnas una columna de dos unos mientras se pueda (hasta que alguno de los dos elementos de cada columna sea 0).
  - Se hace un bucle para todos los divisores primos de n excluyendo el dos (el de los caminos inversos) que se tiene en cuenta en cada paso:
  - Para cada p divisor primo de n se realiza el siguiente tratamiento (elección de los caminos más cortos):

  * Se reescribe la cadena en forma de matriz circular por filas $2p \times N/2p$. Es decir, la fila siguiente a la de índice $2p-1$ es la de índice 0 y la fila anterior a la de índice 0 es $2p-1$.
  * Para cada columna de la matriz se hace el siguiente tratamiento (recuérdese el aspecto circular de cada columna, la búsqueda de las cadenas debe tomar como casilla siguiente a la $2p-1$ -ésima la primea de todas (de índice 0)).
    - BUSCAR CADENA: Se busca la primera secuencia con el siguiente aspecto 1010...01 y que contenga $\frac{p+1}{2}$ dígitos 1 (es en total una cadena de $\frac{p+1}{2} + \frac{p+1}{2} - 1 = \frac{p+1+p+1-2}{2} = p$ dígitos). Si no hay ninguna se va a FIN p:
    - La cadena es sustituída por otra en la que se intercambian ceros y unos, es decir una cadena de p dígitos 010...010 que tendrá en total $\frac{p+1}{2} - 1$ dígitos 1 (uno menos que antes) y $\frac{p+1}{2}$ dígitos 0 (uno más que antes).
    - Se vuelve al punto con etiqueta "BUSCAR CADENA"
  * FIN p: Se elimina las parejas de vectores opuestos: para ello se reescribe la cadena en forma de matriz $2 \times N/2$ y se resta por columnas una columna de dos unos mientras se pueda (hasta que alguno de los dos elementos de cada columna sea 0).

- Si n es impar.
  - Se hace un bucle para todos los divisores primos de n excluyendo el número 2.
  - Para cada p divisor primo de n se realiza el siguiente tratamiento (eliminación de ciclos de p lados):
    * Se reescribe la cadena en forma de matriz $p \times N/p$.
    * Para cada columna de la matriz se hace el siguiente tratamiento.
      - Se elimina columnas compuestas sólo por dígitos 1. Para ello se resta por columnas, columnas formadas por p dígitos 1 hasta que alguna casilla llegue al valor 0.

Ejemplo

En la figura 6.c.11 se muestra un ejemplo de cálculo de la representación canónica. Se detalla cada simplificación:

- Primero se presenta un camino tal y como sería obtenido directamente de la interpretación de una cadena. La tupla $(3,0,2,0,1,0)$ contiene los incrementos de ángulo de sus tramos
- Después se presenta el mismo camino ordenado. El vector de ángulos sería el mismo.
- El tercer gráfico elimina ciclos de tres lados. Se obtiene la tupla $(2,0,1,0,0,0)$
- El último sustituye un camino de dos tramos entre dos vértices de un triángulo por otro de uno solo que representa un camino posible y más corto. La representación canónica es $(1,1,0,0,0,0)$.

- En la misma figura también se muestran los vectores permitidos por la interpretación gráfica.

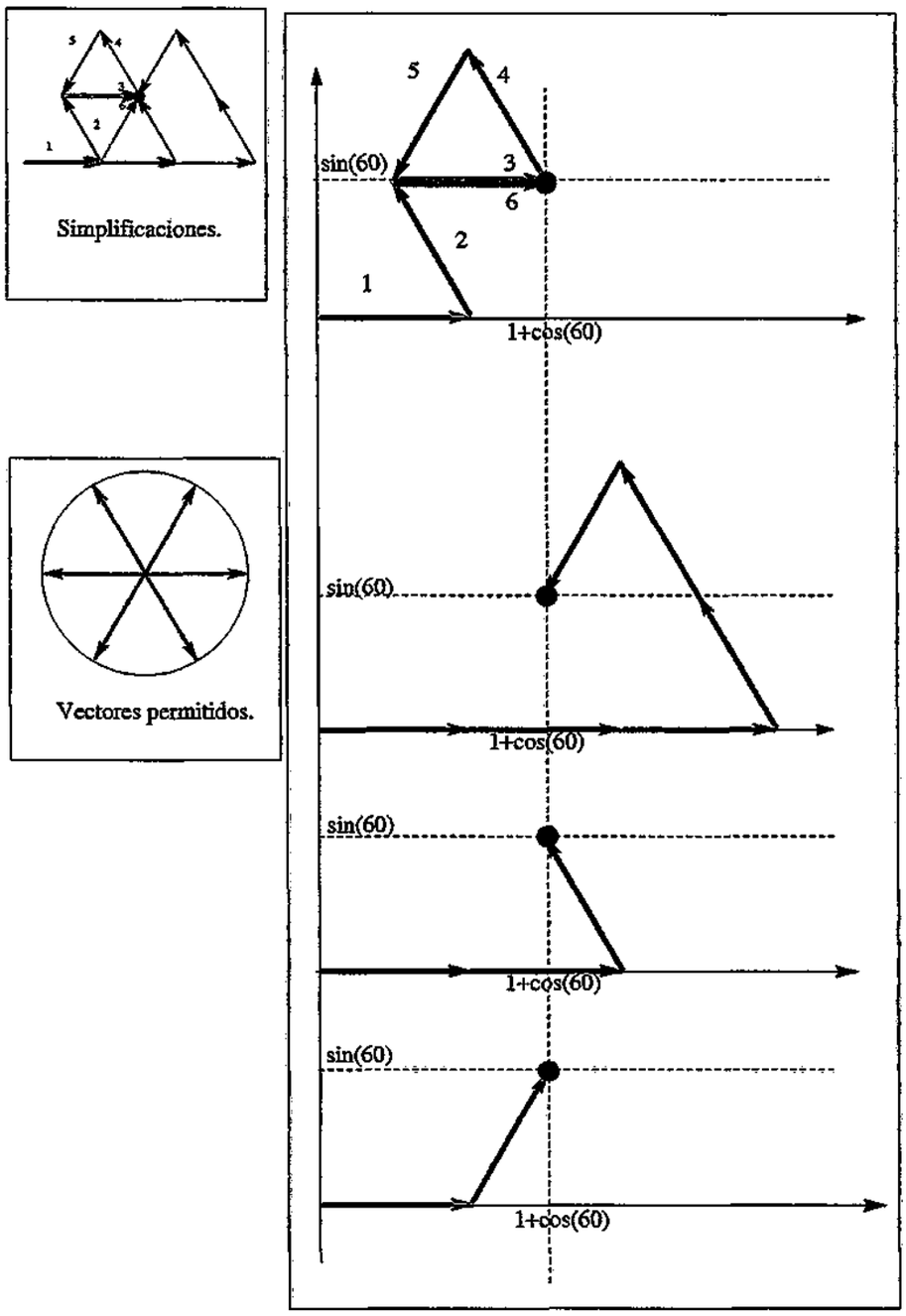


Figura 6.c.11: Ejemplo de obtención de representación canónica.

6.18 Definición

La *interpretación gráfica vectorial de un sistema L* se define de la siguiente manera:

Sea Σ el alfabeto del sistema S.

Se define formalmente interpretación vectorial como la función.

$$G_V : \Sigma \rightarrow \{0,1\} \times \Re^2$$

De forma que

$$G_V(s) = (visibilidad, v_x, v_y)$$

Donde

- $visibilidad = 1$, significa que el vectora asociado al símbolo s es visible (0 en caso contrario).
- (v_x, v_y) es el extremo del vector asociado al símbolo s (supuesto origen (0,0)).

6.19 Definición

El *conjunto de vectores asociado a la interpretación gráfica vectorial de un sistema L* se define según la siguiente expresión:

$$V(G_v) = \{(v_x, v_y) | \exists s \in \Sigma, , G_V(s) = (visibilidad, v_x, v_y)\}$$

6.20 Definición

El *conjunto de módulos asociado a la interpretación gráfica vectorial de un sistema L* se define según la siguiente expresión:

$$M(G_v) = \{m \in \Re | \exists s \in \Sigma, , G_V(s) = (visibilidad, v_x, v_y) \wedge |(v_x, v_y)| = m\}$$

6.21 Notación

El módulo asociado a un símbolo por una interpretación gráfica vectorial se representará de la siguiente forma:

$$m(s) = |(v_x, v_y)| , , G_V(s) = (visibilidad, v_x, v_y) \, \forall visibilidad \in \{0,1\}$$

6.22 Definición

El *conjunto de ángulos asociado a la interpretación gráfica vectorial de un sistema L* se define según la siguiente expresión:

$$\Theta(G_v) = \left\{\phi \in \Re | \exists s \in \Sigma, , G_V(s) = (visibilidad, v_x, v_y) \wedge \arctan\left(\frac{v_x}{v_y}\right) = \phi\right\}$$

6.23 Notación

El ángulo asociado a un símbolo por una interpretación gráfica vectorial se representará de la siguiente forma:

$$\phi(s) = \arctan\left(\frac{v_x}{v_y}\right), , G_V(s) = (visibilidad, v_x, v_y) \, \forall visibilidad \in \{0,1\}$$

6.24 Algoritmo para el cálculo de $\Theta(G_v)$ y $M(G_v)$

El conjunto $V(G_v)$ puede expresarse en coordenadas polares o en coordenadas cartesianas indistintamente:

$V(G_v) = \{(v_x, v_y) \,|\exists s \in \Sigma, , G_V(s) = (visibilidad, v_x, v_y)\} = \{(m, \phi) \,|\, \exists s \in \Sigma, , G_V(s) = (visibilidad, v_x, v_y) \wedge m = |(v_x, v_y)| \wedge \phi = \arctan\left(\frac{v_x}{v_y}\right)\}$

Y por tanto los conjuntos estudiados se pueden construir de la siguiente manera:

$\theta \in \Theta(G_v) \Leftrightarrow \exists m \in \Re | (m, \theta) \in V(G_V)$

$m \in M(G_v) \Leftrightarrow \exists \theta \in \Re | (m, \theta) \in V(G_V)$

6.25 Definición, compatibilidad entre representación gráfica vectorial y sistemas de Lindenmayer

Se dice que una *interpretación gráfica vectorial* G_V es *compatible con un sistema de Lindenmayer* $S \Leftrightarrow alfabeto(S) \subseteq dom(G_V)$, donde $dom(f)$ hace referencia al dominio de una función f.

6.26 Definición, gráfico vectorial de Lindenmayer

Se llama *gráfico vectorial de Lindenmayer* a aquél cuya interpretación gráfica es vectorial.

6.27 Conjunto de números reales relacionados racionalmente

6.27.1 Definición

Un conjunto de números reales se dice que *está racionalmente relacionado* si y sólo si el cociente de cualaesquiera dos de ellos es un número racional.

Formalmente:

$$V \subset \Re \text{ racionalmente relacionado} \Longleftrightarrow \forall v_1, v_2 \in V \Rightarrow \frac{v_1}{v_2} \in Q$$

6.27.2 Lema

Cualquier subconjunto de Q está relacionado racionalmente.

6.27.3 Justificación.

Resulta trivial por la propia definición de conjunto relacionado racionalmente. Si todos los elementos de un conjunto son racionales. Cualquier cociente formado con elementos del conjunto será un cociente de racionales.

6.27.4 Lema

Todo conjunto en el que haya al menos un elemento racional distinto de cero y otro irracional no está relacionado racionalmente.

6.27.5 Justificación

Resulta trivial por la propia definición de conjunto relacionado racionalmente. No es posible obtener un número racional como cociente de un número irracional entre otro racional. Por tanto, cualquier pareja formada con estos dos elementos tendría cociente irracional y el conjunto no estaría racionalmente relacionado.

6.27.6 Teorema, caracterización de partes de $\Re$ relacionados racionalmente

Enunciado

Un conjunto finito de números reales está racionalmente relacionado si y sólo si todos los elementos son múltiplos enteros del mismo factor real.

Formalmente

$$V \subset \Re \text{ racionalmente relacionado} \Leftrightarrow \exists r \in \Re | \forall v \in V, v = mr, , m \in Z$$

Demostración

Demostración directa. Sea $V = \{v_1, v_2, ..., v_n\}$. Al ser V finito, siempre podrá ser definido por extensión.

Como V está racionalmente relacionado el cociente de cualesquiera dos de sus elementos será un número racional. Tomando, por ejemplo, el elemento v_1, el cociente de todos los elementos de V y v_1 también tendrá esa propiedad:

$$\forall v_i \in V \; \exists p_i, q_i \in Z \; | \frac{v_i}{v_1} = \frac{p_i}{q_i}$$

Esto permite expresar todos los elementos de V de la siguiente manera

$$\forall v_i \in V \; v_i = p_i \frac{v_1}{q_i} \tag{1}$$

Para cada elemento de V existirá un denominador q_i en general distinto. Para expresar cada elemento de V como múltiplo entero del mismo factor interesa obtener el denominador común a todas esas fracciones. Para ello se calcula el *mcm* de todos los denominadores.

$$\text{Sea } q = mcm_{i\in}\,(q_i) \Rightarrow \text{(por definición de mínimo común múltiplo)}$$
$$\forall i \in \{1,..,n\}\, \exists k_i \in Z | q = k_i q_i \Rightarrow \forall i \in \{1,..,n\}\, \exists k_i \in Z | \tfrac{q}{k_i} = q_i$$

Se puede ahora introducir en la expresión (1) la expresión (2)

$$\forall v_i \in V\ \exists p_i, q_i, k_i \in Z\ | v_i = p_i k_i \tfrac{v_1}{q}$$

Llamemos r a la fracción del producto de la expresión (3)

$$r = \tfrac{v_1}{q}$$

En virtud del desarrollo anterior se podrá afirmar:

$$\forall v_i \in V\ \exists p_i, k_i \in Z, r \in \Re\ | v_i = p_i k_i r$$

Como se quería demostrar.

Demostración inversa

Si $\exists r \in \Re | \forall v \in V, v = mr, , m \in Z \Rightarrow$

Es suficiente reescribir cualquier elemento de V como el producto dél entero y r para obtener el resultado: al dividir se puede simplificar r y lo que queda es racional.

$\forall v_i, v_j \in V \exists m_i, m_j \in Z, , v_i = m_i r \wedge v_j = m_j r$ (r definido antes)$\Rightarrow$

$\forall v_i, v_j \in V \frac{v_i}{v_j} = \frac{m_i r}{m_j r} = \frac{m_i}{m_j} \in Q$ (ya que q_i y $q_j \in Z$ y por definición de Q).$\Leftrightarrow$ V está racionalmente relacionado.

Esta última expresión es la definición de conjunto racionalmente relacionado.

Ejemplo

- Los siguientes conjuntos de números no están relacionados racionalmente:
  - $V = \left\{2\pi, \frac{7}{4}, \frac{5}{3}\pi\right\}$, la mezcla de valores racionales $(\frac{7}{4})$ e irracionales (los demás) hace que sea imposible que algunos cocientes sean racionales, en particular todos aquellos que tengan un elemento racional y el otro irracional.
  - $V = \left\{2\pi, \frac{7}{4}e, \frac{5}{3}\pi, \frac{13}{3}\sqrt{2}\right\}$, la mezcla de valores irracionales incomparables $(e, \pi, \sqrt{2})$ hace imposible que algunos cocientes sean racionales.
- Los siguientes conjuntos de números reales están relacionados racionalmente:
  - $V = \{2, 7, 5\}$. En este caso, $r = 1$.
  - $V = \left\{2\pi, \frac{7}{4}\pi, \frac{5}{3}\pi\right\}$ (podrían ser los ángulos permitidos en una interpretación gráfica de tipo tortuga). En este caso $r = \frac{\pi}{12}$ y el conjunto V puede expresarse finalmente de la siguiente manera:

$$V = \left\{24\tfrac{\pi}{12}, 21\tfrac{\pi}{12}, 20\tfrac{\pi}{12}\right\}$$

– $V = \left\{2, \frac{7}{4}, \frac{5}{3}\right\}$. En este caso $r = \frac{1}{12}$ y el conjunto V puede expresarse finalmente de la siguiente manera:

$$V = \left\{24\frac{1}{12}, 21\frac{1}{12}, 20\frac{1}{12}\right\}$$

6.28 Gráficos vectoriales de Lindenmayer relacionados racionalmente

6.28.1 Definición

Se dice de un esquema VGD0L S de un gráfico vectorial de Lindenmayer que está racionalmente relacionado (y se llamará en adelante esquema RRVGD0L) si y sólo si el conjunto de los módulos y el conjunto de los ángulos de los vectores de la interpretación gráfica G_v ($\Theta(G_v)$ y $M(G_v)$) están racionalmente relacionados.

A partir de ahora se mencionará indistintamente esquema, sistema o gráfico de Lindenmayer relacionado racionalmente cuando el contexto elimine las posibles ambigüedades.

6.28.2 Corolario

En cualquier esquema RRVGD0L hay dos números reales (r y α) tales que:

- Todos los ángulos de los vectores de la interpretación gráfica son múltiplos enteros positivos de α.
- Todos los módulos de los vectores de la interpretación gráfica son mútiplos enteros positivos de r.

Capítulo 7

Teorema de equivalencia entre dos representaciones gráficas de sistemas L

7.1 Justificación para el estudio de la equivalencia entre los dos tipos de representaciónes.

Entre las razones que hacen interesante encontrar un sistema D0L con interpretación gráfica de tipo vectorial (VGD0L) gráficamente equivalente a otro con interpretación gráfica de tipo tortuga (TGD0L) y viceversa, se puede citar las siguientes:

- Los gráficos vectoriales son habitualmente más rápidos que los gráficos de tipo tortuga. Encontrar un VGD0L gráficamente equivalente a un TGD0L determinado puede resultar interesante por razones de eficiencia en el tratamiento informático.
- Los gráficos de tipo tortuga son más flexibles que los gráficos de tipo vectorial. Un TGD0L gráficamente equivalente a un VGD0L determinado puede resultar más adecuado si se necesita manipular la figura, por ejemplo rellenando áreas cerradas por ella o coloreando diferentes secciones del fractal.

El objeto de los siguientes párrafos es mostrar un teorema de equivalencia entre dos subconjuntos interesantes de las dos familias presentadas previamente. Un resultado más general para la obtención de esquemas TGD0L gráficamente equivalentes a cualquier esquema VGD0L, y viceversa, no es aún posible.

7.2 Teorema 1, obtención de sistemas y esquemas L con interpretación tortuga gráficamente equivalente a otro con interpretación vectorial

Para todo sistema AITGD0L que represente a un fractal con la interpretación gráfica tortuga usual y con un ángulo $\alpha = \frac{2k\pi}{n}$ existe un sistema RRVGD0L gráficamente equivalente.

Para todo esquema AITGD0L que represente a un conjunto de fractales con la interpretación gráfica usual y con $\alpha = \frac{2k\pi}{n}$ existe un esquema RRVGD0L gráficamente equivalente.

7.3 Demostración

7.3.1 Descripción informal

Se proporciona un algoritmo que, dado un esquema AITGD0L obtiene otro RRVGD0L gráficamente equivalente respecto al fractal. Puede aplicarse el mismo algoritmo a un sistema AITGD0L. A partir de su axioma se obtiene una cadena con interpretación gráfica vectorial que genera la misma curva. El axioma y el esquema obtenidos forman un sistema RRVGD0L gráficamente equivalente.

El proceso tiene los siguientes pasos:

Símbolos del nuevo alfabeto: símbolos comunes

Sea Σ el alfabeto del esquema AITGD0L de partida y Σ' el alfabeto del esquema RRVGD0L asociado. Para la construcción del alfabeto del esquema RRVGD0L sólo se tiene en cuenta los símbolos de Σ distintos de +, −, (y). También se tiene en cuenta el valor de n (recuérdese que en la interpretación gráfica del sistema L de partida hay un ángulo básico $\alpha = \frac{2k\pi}{n}$). Se ha señalado ya que el número de distintos desplazamientos posibles con esta interpretación gráfica es n. En el nuevo sistema RRVGD0L es necesario definir de forma explícita un símbolo que tenga un vector asociado para cada desplazamiento posible en el sistema de partida. Por tanto, para cada elemento s de $\Sigma - \{+.-,(,)\}$ se añadirá n símbolos a Σ'. Cada uno ellos se utilizará en lugar de s en cada una de las posibles inclinaciones con las que aparezca en las palabras del sistema de partida.

La figura 5.t1.a muestra dos ejemplos. En ella aparecen los símbolos que en Σ' estarán asociados al símbolo F de Σ. Todos los vectores tienen módulo 1, si tomamos como unidad el desplazamiento de la tortuga asociado a los símbolos de dibujo y movimiento.

- En la gráfica de la izquierda se supone una interpretación gráfica con n=8 y k=1; por tanto $\alpha = 45$.
- En la de la derecha se supone una interpretación gráfica con n=5 y k=1; por tanto $\alpha = 72$.

Es aconsejable utilizar una representación que recuerde cuántos incrementos del ángulo α tiene asociados cada símbolo de Σ'. Se utilizará a lo largo de la demostración la siguiente notación

$s'(i)$ donde $s \in \Sigma$ e $i \in Z_n$

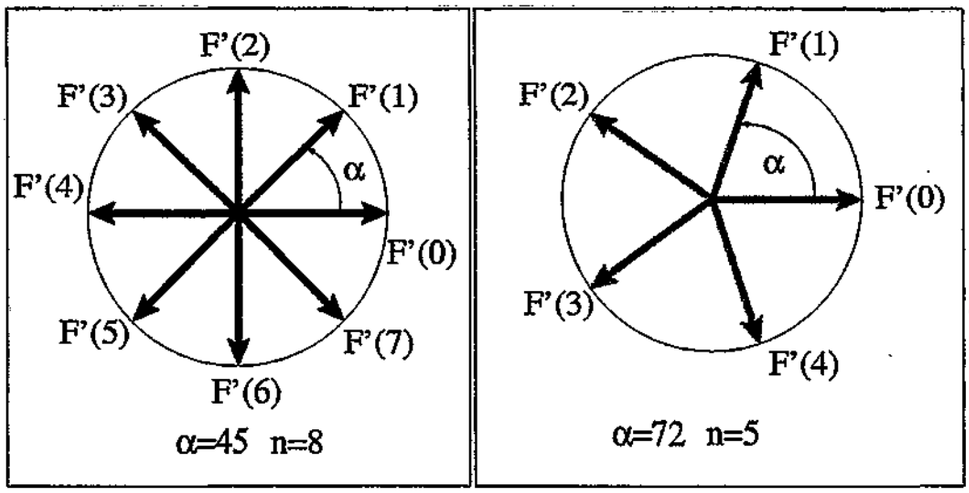


Figura 5.t1.a: Sigma prima.

Símbolos del nuevo alfabeto: símbolos de retorno para alfabetos con paréntesis.

Cuando en el alfabeto de partida (Σ) hay paréntesis es posible representar ramificaciones al aplicar la interpretación gráfica de tipo tortuga a las palabras del sistema L . En la figura 5.t1.b se muestra un ejemplo de este tipo de ramas: el recorrido de color rojo es una ramificación.

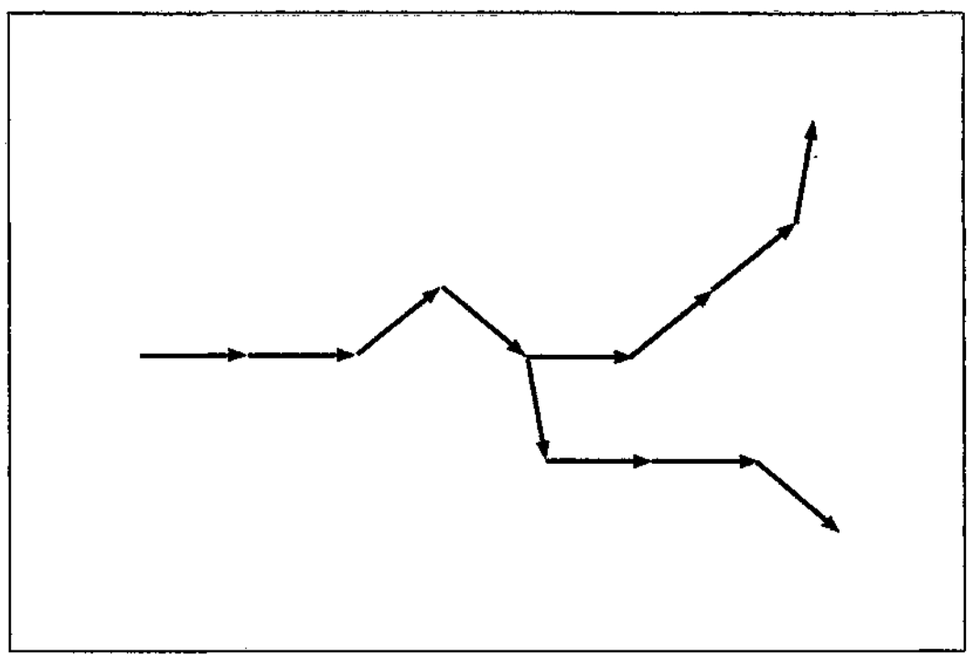


Figura 5.t1.b : Ejemplo de rama.

Para poder representar una rama en el sistema L con interpretación vectorial equivalente, una vez se haya terminado cada rama es necesario volver al punto en que ésta se originó..

Ya se mostró previamente cómo el conjunto de vectores asociados a la interpretación tortuga de un sistema L contiene vectores opuestos si y sólo si el valor de n tal que $\alpha = \frac{2k\pi}{n}$ es par. Por tanto, en sistemas L con interpretación tortuga con n par siempre se puede construir la rama opuesta. Bastaría llevar cuenta de cuáles son las parejas de opuestos y formar con ellos la cadena opuesta a la rama terminada, como muestra la figura 5.t1.c.

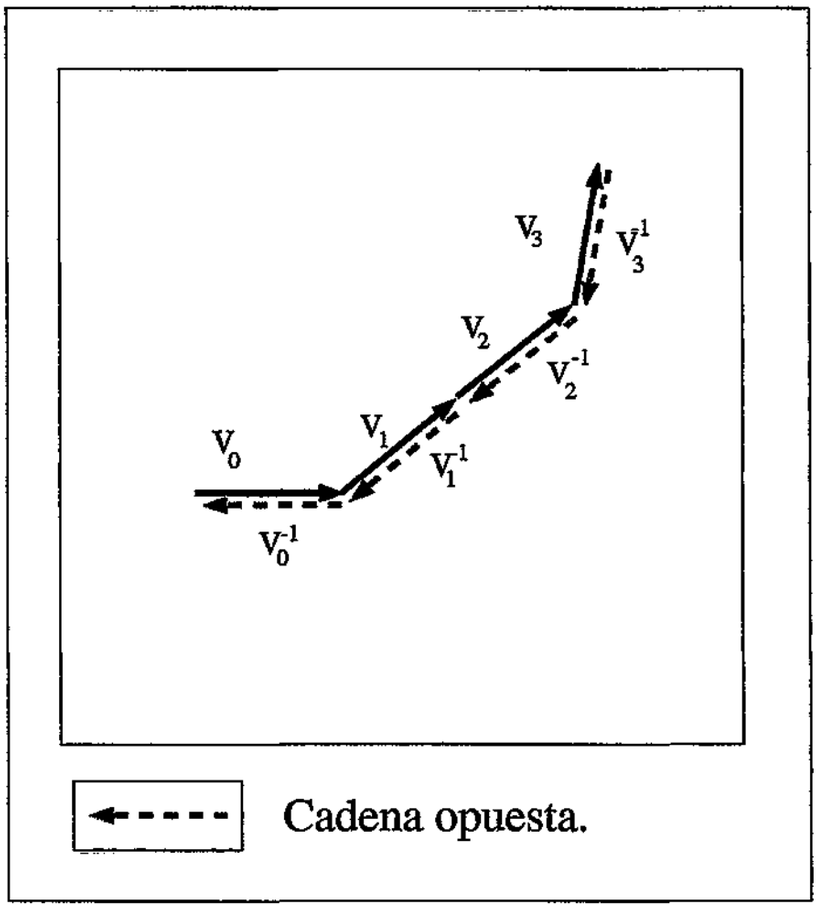


Figura 5.t1.c: Ejemplo de cadena opuesta.

Para resolver este problema existen dos posibilidades.

- Se ha mostrado previamente que el ciclo más corto que se puede conseguir con los vectores de una interpretación gráfica de un sistema L ($V(G_T)$) tiene p lados cuando p es el más pequeño de los divisores primos de n. También se ha mostrado que se pueden cerrar ciclos

de p lados a partir de cualquier elemento de $V(G_T)$. En el caso de que n sea impar, se puede encontrar una cadena que represente un camino opuesto a partir de símbolos asociados a los vectores de $V(G_T)$ que cierran esos ciclos como se aprecia en la gráfica siguiente.

La interpretación gráfica de la figura 5.t1.d utiliza n=9 que es impar. El divisor primo más pequeño de 9 es 3 por lo que el ciclo mínimo es de 3 lados. Se muestra una cadena y el camino opuesto formado al cerrar cíclos mínimos (triángulos) que tienen como uno de sus lados cada tramo del camino. Según esta opción, para tratar las ramas se tendría que tener en cuenta el resto del ciclo mínimo que se puede formar a partir de cada posible vector y tras finalizar cada rama insertar la cadena cuya interpretación fuera un camino de retorno.

Todos los vectores utilizados para deshacer el camino tendrían que tener visibilidad nula.

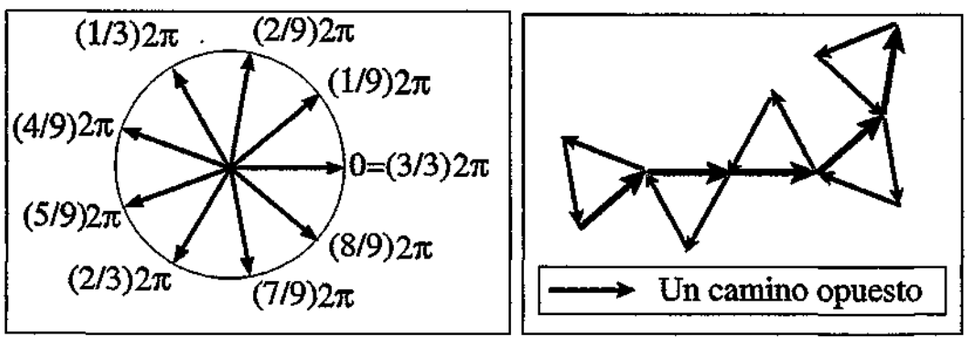


Figura 5.t1.d: Un camino opuesto.

- También se ha mostrado previamente que la interpretación gráfica cuyo conjunto de vectores es el más pequeño posible que incluye a todos los de otra interpretación con n impar y a todos sus opuestos es precisamente la que utiliza en lugar de n el valor 2n. Por lo tanto también es posible en estos casos partir de una interpretación tortuga de 2n incrementos de ángulo. La figura 5.t1.e muestra un ejemplo.

  A la izquierda se muestra los 5 símbolos asociados al símbolo F (supuesto que $F \in \Sigma$ en el sistema de partida y que n=5). A la derecha se muestra los 10 símbolos correspondientes al supuesto n=10. Es fácil identificar las parejas de vectores opuestos. Los símbolos correspondientes a los vectores opuestos se nombrarán insertando una "doble comilla" entre el símbolo y la inclinación.

  En este caso la visibilidad de los vectores que forman la cadena opuesta es indiferente.

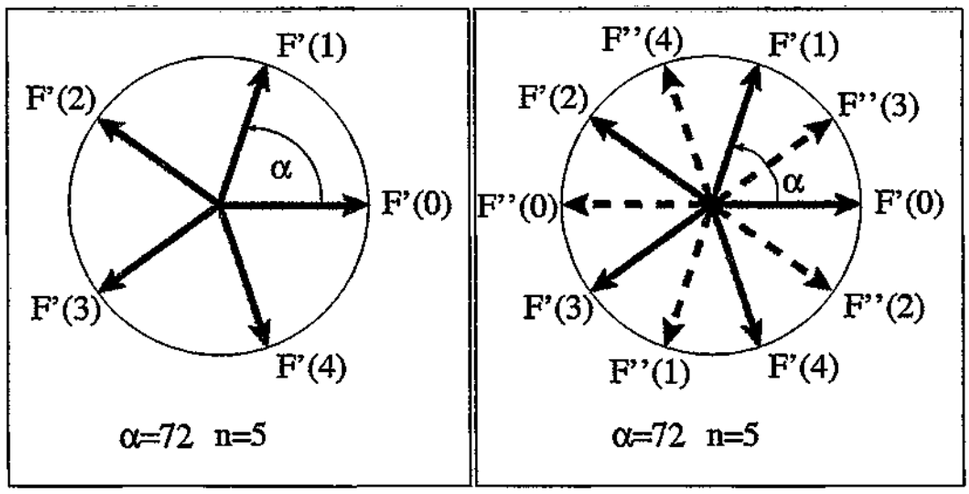


Figura 5.t1.e Vectores necesarios para cadenas opuestas con n impar.

Esta última ha sido la opción que se ha elegido para la demostración del teorema. Cuando en el alfabeto de partida haya paréntesis, a los símbolos asociados a cada inclinación se le añadirán los opuestos. Esta operación se realizará, para simplificar la exposición, de manera independiente a que n sea par o impar. Aunque en el primer caso el número de símbolos podría reducirse a la mitad, pero sólo pretendemos demostrar la equivalencia, no obtener el sistema equivalente más sencillo posible.

Construcción de Σ': recapitulación

$$\Sigma' = \left\{ \begin{array}{ll} \{s'(i),,i \in Z_n \forall s \in alfabetos(G_T)\} \cup & \\ \{s''(i),,i \in Z_n \forall s \in alfabetos(G_T)\} & si \quad \{(,)\} \subseteq \Sigma' \\ & \\ \{s'(i),,i \in Z_n \forall s \in alfabetos(G_T)\} & si \quad \neg\{(,)\} \subseteq \Sigma' \end{array} \right\}$$

Tratamiento de las cadenas: el axioma

A cada símbolo de $\Sigma - \{+,-,(,)\}$ le corresponde una familia de símbolos en Σ' (dos si hay paréntesis) y cada uno de ellos tiene asociado el vector correspondiente a una posible inclinación (que está reflejada de forma explícita en el nombre del símbolo en Σ'). En la interpretación gráfica de tortuga, ésta tiene una inclinación en cada momento. Si se interpreta una cadena será sencillo encontrar cuál es el símbolo de Σ' asociado al próxima símbolo de la cadena. Sea s este símbolo:

- Si $s \in \Sigma - \{+,-,(,)\}$ y m es el número de incrementos de α de la inclinación actual, el símbolo asociado en Σ' es $s'(m)$
- Si $s \in \{+,-\}$ habrá que modificar m de forma adecuada.

- Si $s \in \{(,)\}$ habrá que realizar el tratamiento de ramificación que se ha descrito en los párrafos anteriores.

Para obtener la cadena asociada en el nuevo sistema es suficiente recorrer la cadena de partida y concatenar los símbolos asociados en la manera que se ha especificado en la lista anterior.

A continuación se muestra un ejemplo de tratamiento de una cadena (que podría ser el axioma de un sistema L).

Sea un sistema L con interpretación gráfica de tortuga, n = 9 y $\alpha = 40$ grados. La cadena del ejemplo es

$$FF+F--F-F++FF-F$$

Se supone una inclinación inicial de la tortuga de 80 grados ($m = 2$). La cadena es interpretada a partir de esa inclinación. Por tanto, la cadena asociada se obtiene concatenando los símbolos deducidos de la siguiente forma:

- F' (2) puesto que el primer símbolo encontrado al tratar la cadena es F y la inclinación es de 2 incrementos de α. Queda por tratar la cadena $F+F--F-F++FF-F$.
- F' (2) puesto que el primer símbolo encontrado al tratar la cadena pendiente de tratamiento es F y la inclinación sigue siendo de 2 incrementos de α. Queda por tratar la cadena $+F--F-F++FF-F$.
- F' (3) puesto que el primer símbolo encontrado al tratar la cadena pendiente de tratamiento es + que supone considerar un incremento unitario en la inclinación actual (m pasa a valer 3) y el siguiente símbolo es F. Queda por tratar la cadena $--F-F++FF-F$.
- F' (1) puesto que los primeros símbolos encontrados al tratar la cadena pendiente de tratamiento son−− que suponen considerar un decremento de valor 2 en la inclinación actual (m pasa a valer 1) y el siguiente símbolo es F. Queda por tratar la cadena $-F++FF-F$.
- F' (0) puesto que el primer símbolo encontrado al tratar la cadena pendiente de tratamiento es − que supone considerar un decremento unitario en la inclinación actual (m pasa a valer 0) y el siguiente símbolo es F. Queda por tratar la cadena $++FF-F$.
- F' (2) puesto que los primeros símbolos encontrados al tratar la cadena pendiente de tratamiento son ++ que suponen considerar un incremento de valor 2 en la inclinación actual (m pasa a valer 2) y el siguiente símbolo es F. Queda por tratar la cadena $F-F$.
- F' (2) puesto que el primer símbolo encontrado al tratar la cadena pendiente de tratamiento es F y la inclinación sigue siendo de 2 incrementos de α. Queda por tratar la cadena $-F$.
- F' (1) puesto que el primer símbolo encontrado al tratar la cadena pendiente de tratamiento es − que supone considerar un decremento unitario en la inclinación actual (m pasa a valer 1) y el siguiente símbolo es F. Se ha completado el tratamiento de la cadena.

La cadena asociada es, por tanto (véase la figura 5.t1.f):

$$F'(2)\,F'(2)\,F'(3)\,F'(1)\,F'(0)\,F'(2)\,F'(2)\,F'(1)$$

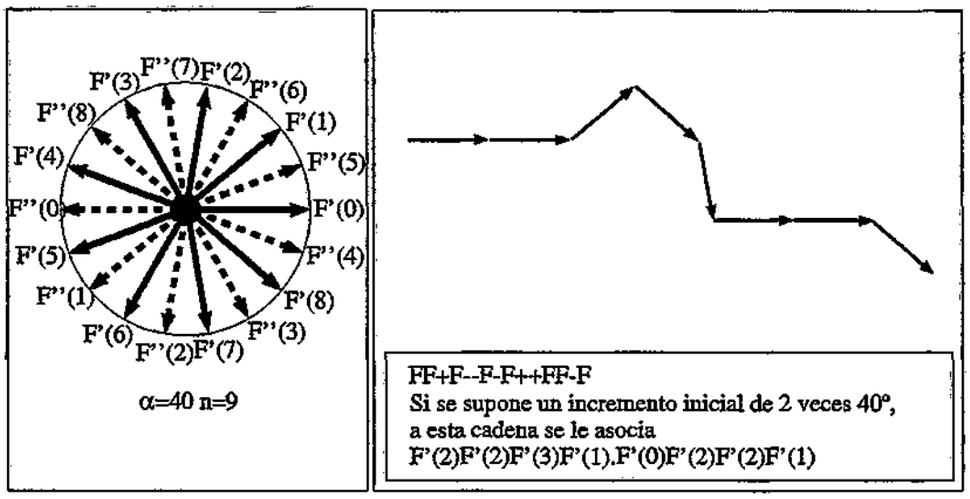


Figura 5.t1.f: Transformación de una cadena.

A continuación (véase la figura 5.t1.g) se muestra otro ejemplo de tratamiento de cadena, obtenido al añadir una rama a la cadena del ejemplo anterior. La rama nace en el cuarto símbolo dibujable de la cadena de partida, donde la inclinación es de 1 incremento de α.

Se ha indicado anteriormente que el tratamiento de toda rama necesita dos fases:

- Tratamiento de la rama como cadena normal.
- Tratamiento del camino inverso.

La rama es la cadena $+F + FF + F$.

Como la inclinación actual es 1, siguiendo un razonamiento análogo al anterior se obtendrá como cadena asociada en el sistema final $F'(2)\,F'(3)\,F'(3)\,F'(4)$.

Se ha indicado en los puntos precedentes que para construir un camino de retorno se optará por duplicar el número de vectores. Para ello, se considera en este caso un valor efectivo de n=18. Las parejas de vectores opuestos están identificadas en la figura. Para volver al punto de partida será necesario concatenar, una vez terminada la rama, la cadena opuesta.

$$(F'(2)\,F'(3)\,F'(3)\,F'(4))^{-1} = (F'(4))^{-1}\,(F'(3))^{-1}\,(F'(3))^{-1}\,(F'(2))^{-1} = F''(4)\,F''(3)\,F''(3)\,F''(2)$$

Al insertar, tras el cuarto símbolo de la cadena resultado del ejemplo anterior, la concatenación de las dos cadenas asociadas a la rama, se obtiene finalmente:

$$F'(2)\,F'(2)\,F'(3)\,F'(1)\,F'(2)\,F'(3)\,F'(3)\,F'(4)$$
$$F''(4)\,F''(3)\,F''(3)\,F''(2)\,F'(0)\,F'(2)\,F'(2)\,F'(1)$$

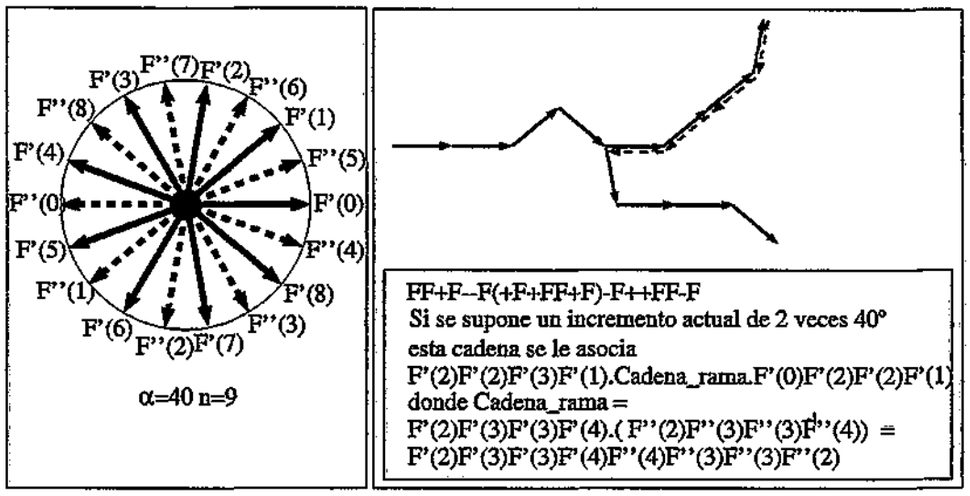


Figura 5.t1.g: Transformación de una cadena con rama.

Tratamiento de las cadenas: reglas de producción

Las observaciones realizadas en los párrafos precedentes para el axioma pueden aplicarse a cualquier cadena generada por el sistema L de partida. Las reglas en el sistema de partida asocian una cadena (parte derecha) a cada símbolo de $\Sigma - \{+, -, (,)\}$ (parte izquierda). En el nuevo sistema nos encontramos con una familia (o dos) de símbolos $(s'(i),, i \in Z_n$ y $s''(i),, i \in Z_n)$, que representan posibles inclinaciones en las que el símbolo puede presentarse. Será necesario añadir al sistema equivalente una copia de cada regla para cada una de las posibles inclinaciones iniciales como se muestra en la figura 5.t1.h.

Sea un sistema L sin paréntesis, con una interpretación gráfica con n=5. A la regla del sistema tortuga de partida

$$F ::= F - F - F + +F$$

es necesario asociarle cinco (n=5) reglas distintas en el sistema final. El conjunto de símbolos asociados a F en Σ'es $\{F'(0), F'(1), F'(2), F'(3), F'(4)\}$. Habrá una regla que tome como parte izquierda cada uno de estos símbolos y como parte derecha la cadena derivada de $F - F - F + +F$. La inclinación inicial de la cadena se toma de la parte izquierda de la regla.

Se obtendrá:

$$F'(0) ::= F'(0)\, F'(4)\, F'(3)\, F'(1)$$

$$F'(1) ::= F'(1)\, F'(0)\, F'(4)\, F'(2)$$

$$F'(2) ::= F'(2)\, F'(1)\, F'(0)\, F'(3)$$

$$F^{l}(3) ::= F^{l}(3)\,F^{l}(2)\,F^{l}(1)\,F^{l}(4)$$

$$F^{l}(4) ::= F^{l}(4)\,F^{l}(3)\,F^{l}(2)\,F^{l}(0)$$

Es fácil observar que las inclinaciones de cada símbolo mantienen la estructura de la cadena de partida.

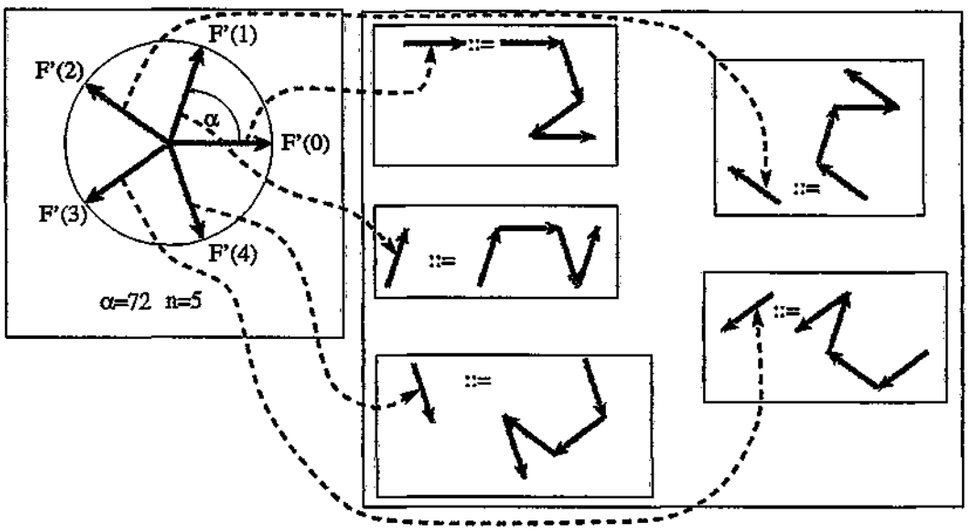


Figura 5.t1.h: Reglas asociadas con cada inclinación.

En el caso $\{(,)\} \subseteq \Sigma$ la segunda familia de símbolos es necesaria,pero el tratamiento es exactamente el mismo, excepto que se utilizará los símbolos $s''(m)$ en lugar de $s'(m)$.

Interpretación gráfica

La interpretación gráfica del sistema final se construye fácilmente de la siguiente forma:

- Los símbolos no dibujables del sistema de partida, que eran ignorados por la interpretación gráfica tortuga, no tienen asociado ningún desplazamiento.

$$s \in \text{alfabeto\_no\_gráfico}(G_T) \Rightarrow G_V(s'(i)) = G_V(s''(i)) = (0,0,0)$$

- Los símbolos asociados en el sistema final a símbolos dibujables y de movimiento en el sistema de partida tienen asociadas las coordenadas cartesianas del desplazamiento que definen: todos tienen el mismo módulo (1) de la interpretación gráfica tortuga de partida. Los de dibujo deberán ser visibles y los de movimiento invisibles.

$$s \in \text{alfabeto\_dibujable}(G_T) \Rightarrow G_V(s'(i)) = (1,\cos(i\alpha),\sin(i\alpha)) \wedge G_V(s''(i)) = (1,-\cos(i\alpha),-\sin(i\alpha))$$

$$s \in \text{alfabeto\_movimiento}(G_T) \Rightarrow G_V(s'(i)) = (0,\cos(i\alpha),\sin(i\alpha)) \wedge G_V(s''(i)) = (0,-\cos(i\alpha),-\sin(i\alpha))$$

7.3.2 Demostración formal

Sea $L = (\Sigma, P, w)$ un sistema AITGD0L.

Se utilizará los siguientes supuestos y notaciones:

- Sea $\Sigma = D \cup M \cup U \cup \{+, -, (,)\}$ el alfabeto, utilizando la notación presentada en el capítulo 6.
- Las reglas de producción son:

  $+ ::= + \in P$

  $- ::= - \in P$

  $(::= (\in P$

  $) ::=) \in P$

  $s ::= x(s) \in P, , x(s) \in \Sigma^* \forall s \in D \cup M \cup U$. Es decir, se identificará mediante $x(s)$ la parte derecha de la regla de producción que tiene como parte izquierda el símbolo s.

Se construye el siguiente gráfico de Lindenmayer (L', G_V) donde $L' = (\Sigma', P', w')$ es un sistema D0L tal que $\forall s \in D \cup M \cup U$

- $s'(i) \in \Sigma' \quad \forall i \in Z_n$.
- $s''(i) \in \Sigma' \quad \forall i \in Z_n$
- $s'(i) ::= C'[x(s), i, 0] \in P'$.
- $s''(i) ::= C''[x(s), i, 0] \in P'$.

Donde $C'[x, i, k] : \Sigma^* \times Z_n^2 \rightarrow \Sigma'^*$se define recursivamente como sigue

- Caso básico: tratamiento de la cadena vacía (y del final de cadena)

  $C'[\lambda, i, k] = \lambda \quad \forall i, k(|\lambda| = 0)$
- Caso básico: tratamiento de cada símbolo de $D \cup M \cup U$.

  $C'[s, i, k] = s'(i + k) \quad \forall s \in D \cup M \cup U$
- Caso recursivo: tratamiento de una cadena que comience por un símbolo de $D \cup M \cup U$.

  $C'[s.y, i, k] = C'[s, i, k].C'[y, i, k] \quad \forall s \in D \cup M \cup U, \forall y \in \Sigma^*$.

- Caso básico: tratamiento de símbolos + y − al final de cadena.

  $C'[+,i,k] = \lambda \quad \forall i,k$

  $C'[-,i,k] = \lambda \quad \forall i,k$

- Caso recursivo: tratamiento de una cadena que comience por un símbolo + o un símbolo −.

  $C'[+.y,i,k] = C'[y,i,k+1] \forall y \in \Sigma^*$

  $C'[-.y,i,k] = C'[y,i,k-1] \forall y \in \Sigma^*$

- Caso recursivo: tratamiento de las ramas (cadenas entre paréntesis).

  $C'[(.x.),i,k] = C'[x,i,k].C''[\text{primer\_nivel}(x),i,k]^{-1} \quad \forall x \in \Sigma^*.$

Donde la función primer nivel fue definida en el capítulo 6 y $C''[x,i,k] : (\Sigma - \{(,)\})^* \times Z_n^2 \rightarrow \Sigma'^*$se define recursivamente como sigue

- Caso básico: tratamiento de la cadena vacía (y del final de cadena)

  $C''[\lambda,i,k] = \lambda \quad \forall i,k(|\lambda| = 0)$

- Caso básico: tratamiento de cada símbolo de $D \cup M \cup U$.

  $C''[s,i,k] = s''(i+k) \quad \forall s \in D \cup M \cup U$

- Caso recursivo: tratamiento de una cadena que comience por un símbolo de $D \cup M \cup U$.

  $C''[s.y,i,k] = C''[s,i,k].C'[y,i,k] \quad \forall s \in D \cup M \cup U, \forall y \in \Sigma^*.$

- Caso básico: tratamiento de símbolos + y − al final de cadena.

  $C''[+,i,k] = \lambda \quad \forall i,k$

  $C''[-,i,k] = \lambda \quad \forall i,k$

- Caso recursivo: tratamiento de una cadena que comience por un símbolo + o un símbolo −.

  $C''[+.y,i,k] = C''[y,i,k+1] \quad \forall y \in \Sigma^*$

  $C''[-.y,i,k] = C''[y,i,k-1] \quad \forall y \in \Sigma^*$

  Obsérvese que C'' es igual que C' excepto que se utiliza s'' en lugar de s' y que no puede haber ramas, ya que sólo se utiliza C'' sobre cadenas sin ramas, es decir, sobre los símbolos incluidos en el primer nivel de paréntesis.

Se construye el axioma del sistema de la siguiente manera:

- $w' = C'[w,0,0]$

La interpretación vectorial G_V del sistema RRVGD0L equivalente se construye de la siguiente manera:

- $G_V(s'(i)) = G_V(s''(i)) = (0,0,0) \quad \forall s \in U, \forall i \in Z_n.$

- $G_V(s'(i)) = (1, \cos(i\alpha), \sin(i\alpha)) \quad \forall s \in D, \forall i \in Z_n.$
- $G_V(s''(i)) = (1, -\cos(i\alpha), -\sin(i\alpha)) \quad \forall s \in D, \forall i \in Z_n.$
- $G_V(s'(i)) = (0, \cos(i\alpha), \sin(i\alpha)) \; \forall s \in M, \quad \forall i \in Z_n.$
- $G_V(s''(i)) = (0, -\cos(i\alpha), -\sin(i\alpha)) \quad \forall s \in M, \forall i \in Z_n.$

El conjunto de símbolos s'' podría eliminarse de L' si $\neg\{(,)\} \subset \Sigma$.

Es evidente, por construcción del algoritmo, que la curva fractal representada por el gráfico de Lindenmayer (L', G_V) es la misma que la representada por (L, G_T).

La segunda parte del teorema, que se refiere a esquemas de Lindenmayer, resulta trivial. Dado un esquema de Lindenmayer con interpretación tortuga, el algoritmo es capaz de construir un esquema de Lindenmayer asociado. Para cada axioma que se utilice con el primer sistema, el algoritmo es capaz de construir un axioma en el segundo, tales que los sistemas L obtenidos son gráficamente equivalentes. Esto demuestra el resultado.

7.3.3 Ejemplo de aplicación del teorema 1

El gráfico de Lindenmayer

$$\left(L = \left(P = \left\{ \begin{array}{c} \Sigma = \{F, +, -\}, \\ F ::= F - F + +F - F, \\ + ::= +, \\ - ::= - \end{array} \right\}, \; w = F + +F + +F \right), \right.$$
$$\left. G_T = (D = \{F\}, M = \Phi, U = \Phi, 60) \right)$$

genera el fractal conocido como la curva de copo de nieve de Koch, cuya quinta derivación se representa en la figura 5.t1.i.

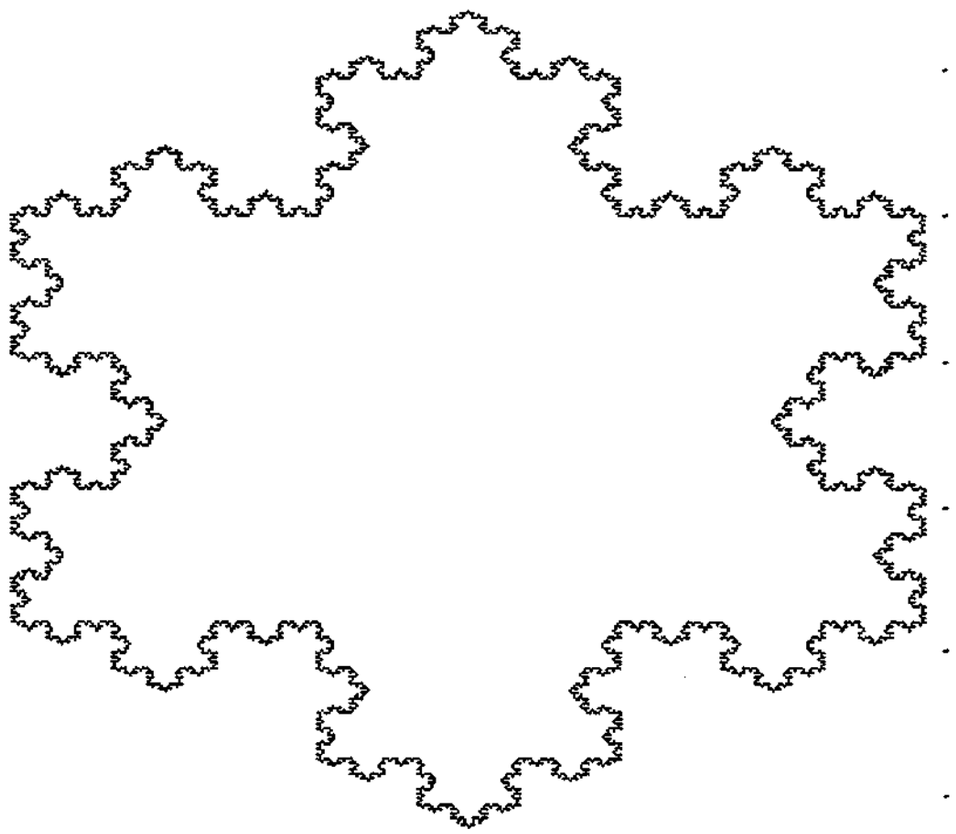


Figura 5.t1.i: Curva de copo de nieve de Koch.

En este caso $n = 6$ y $k = 1$.

Al aplicar el algoritmo a este sistema, se obtiene:

$$\left(L' = \left(\begin{array}{c} \Sigma' = \{A, B, C, D, E, F\}, \\ P' = \left\{ \begin{array}{l} A ::= AFBA, \\ B ::= BACB, \\ C ::= CBDC, \\ D ::= DCED, \\ E ::= EDFE, \\ F ::= FEAF \end{array} \right\}, \\ w' = ACE \end{array} \right), \right.$$

$$\left. G_V(s) = \left\{ \begin{array}{lll} (1,1,0) & si & s = A \\ \left(1, 0.5, \frac{\sqrt{3}}{2}\right) & si & s = B \\ \left(1, -0.5, \frac{\sqrt{3}}{2}\right) & si & s = C \\ (1,-1,0) & si & s = D \\ \left(1, -0.5, -\frac{\sqrt{3}}{2}\right) & si & s = E \\ \left(1, 0.5, -\frac{\sqrt{3}}{2}\right) & si & s = F \end{array} \right\} \right)$$

Donde, por claridad, se ha renombrado los símbolos que genera el algoritmo por claridad, de la siguiente manera:

$$A = F'(0)$$
$$B = F'(1)$$
$$C = F'(2)$$
$$D = F'(3)$$
$$E = F'(4)$$
$$F = F'(5)$$

Puede comprobarse fácilmente que este gráfico de Lindenmayer representa el mismo fractal de la figura 5.t1.i.

7.4 Teorema 2, obtención de sistemas y esquemas L con interpretación vectorial gráficamente equivalente a otro con interpretación tortuga

Para cada sistema RRVGD0L que representa un fractal con interpretación gráfica vectorial, hay un sistema AITGD0L gráficamente equivalente.

Para cada esquema RRVGD0L que representa a un conjunto de fractales con interpretación gráfica vectorial, hay un esquema AITGD0L gráficamente equivalente.

7.5 Demostración

7.5.1 Descripción informal

La demostración se basa en los siguientes puntos:

Aspectos previos

El sistema de partida está relacionado racionalmente, por lo que es posible encontrar los números reales α y r asociados a los dos conjuntos $\Theta(G_V)$ y $M(G_V)$. Para ello es conveniente transformar previamente la interpretación gráfica vectorial en coordenadas cartesianas a coordenadas polares, como se muestra en el ejemplo de la figura 5.t2.a. Estas dos cantidades (α y r) serán respectivamente el ángulo y el módulo de la interpretación gráfica de tipo tortuga del sisetema equivalente.

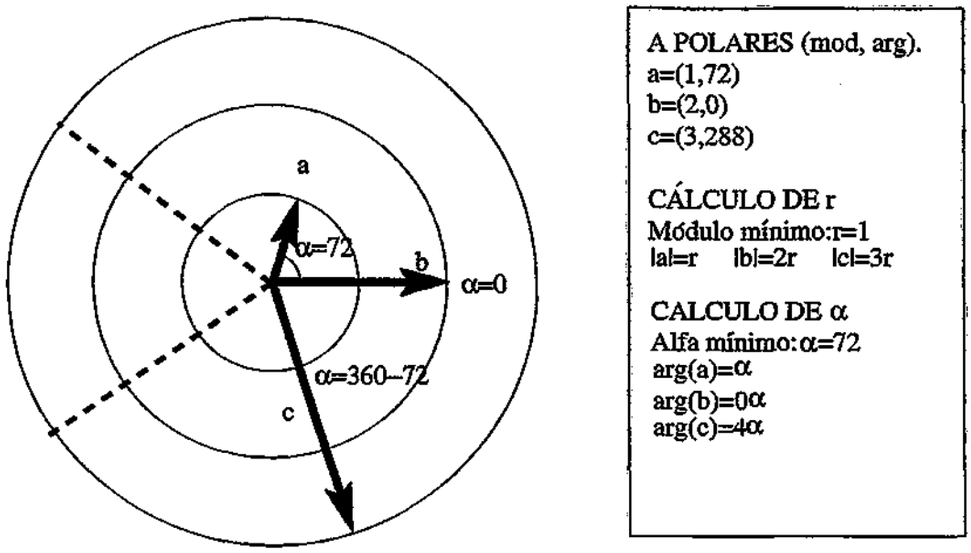


Figura 5.t2.a: Coordenadas polares.

Alfabeto

En el caso de que todos los vectores de la interpretación gráfica tengan el mismo módulo (véase la figura 5.t2.b), es suficiente tener un símbolo en el nuevo alfabeto asociado a cada vector del sistema de partida. A continuación se muestra una interpretación gráfica. Para representar estos símbolos se utilizará la notación siguiente: si s es el símbolo de partida asociado a un vector, s' es el símbolo asociado en el nuevo alfabeto.

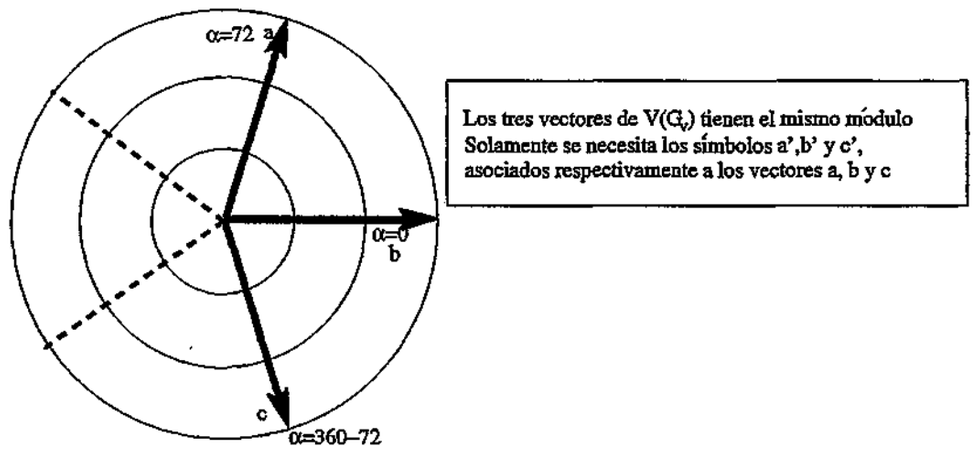


Figura 5.t2.b: Vectores con el mismo módulo.

En caso de que los vectores tengan módulos diferentes ($\exists v_i \in V(G_V) \mid \exists n_i \in \aleph_, , |v_i| = n_i r$

$\wedge\ n_i \neq 1$) es necesario añadir un símbolo adicional por cada uno de los vectores cuyo módulo sea mayor que r, que será repetido hasta alcanzar el módulo adecuado. En el sistema que vamos a construir, todos los desplazamientos tendrán la misma longitud r, que podemos tomar como unidad y coincidirá con el deslazamiento elemental de la tortuga. Se necesitará un total de n_i desplazamientos para alcanzar un desplazamiento igual a $|v_i| = n_i r$. Para representar los símbolos adicionales utilizaremos la notación siguiente: si s es el símbolo del sistema de partida asociado a un vector, s'' es el segundo símbolo asociado en el nuevo alfabeto.

La figura 5.t2.c muestra una interpretación gráfica vectorial con estas características. El valor de r es 1 y por tanto es necesario utilizar un símbolo adicional b" para obtener el módulo correcto del vector b y dos símbolos adicionales c" para obtener el desplazamiento correcto asociado al vector c.

El alfabeto del sistema AITGD0L equivalente se completa con los símbolos habituales en las interpretaciones tortuga ($\{+,-\}$). Obsérvese que, al no haber forma explícita de indicar ramas en la interpretación vectorial (una rama puede conseguirse volviendo a un punto anterior de la gráfica mediante un desplazamiento con visibilidad = 0) el sistema obtenido no necesita paréntesis.

Tratamiento de las cadenas: axioma y reglas

A partir de cada cadena del sistema de partida puede obtenerse otra cadena cuya gráfica será la misma en el sistema objetivo. Para ello es suficiente tener en cuenta las siguientes consideraciones:

- La cadena asociada se obtiene como concatenación de las cadenas obtenidas al tratar cada símbolo de la cadena de partida.
- El sistema de partida es RRVGD0L, es decir, se conoce el valor del ángulo factor mínimo (α) y los factores naturales que multiplicados por él generan el ángulo de cada vector. A cada aparición de un símbolo al que se asocia un vector le corresponde en el sistema objetivo una cadena compuesta por las siguientes partes:
  - Tantos símbolos '+' o '−' como sea necesario para alcanzar la inclinación del vector.
  - El símbolo de tipo s' asociado al vector estudiado en el sistema de partida.
  - Tantos símbolos de tipo s'' asociados al vector estudiado como sean necesarios para alcanzar el módulo del mismo (si el módulo es kr , se añadirá $k-1$ símbolos s'').
  - Una cadena compuesta enteramente por símbolos '+' o '−' opuesta a la del primer punto, que deje la tortuga apuntando en la misma dirección y sentido en el mismo estado en que estaba antes de tratar el símbolo estudiado.
- El tratamiento de las cadenas es el mismo, tanto para obtener el axioma asociado al de partida, como las reglas de producción.

La figura 5.t2.c muestra un ejemplo de tratamiento de cadenas:

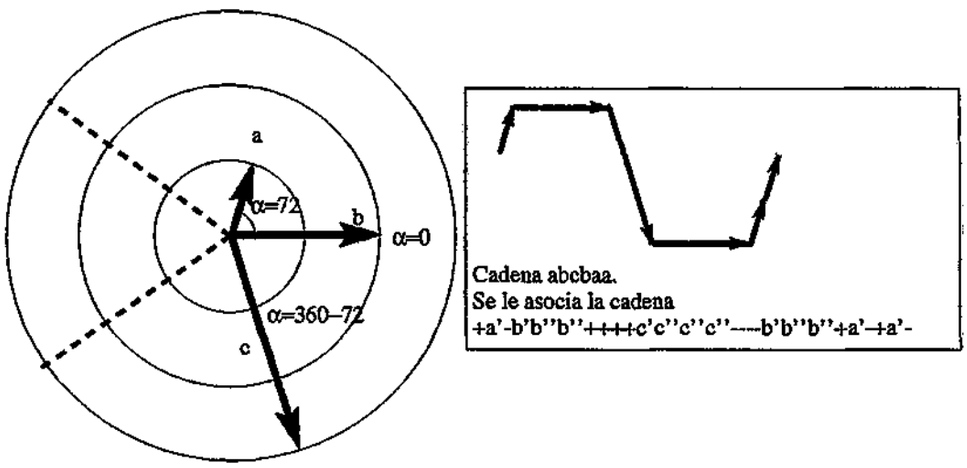


Figura 5.t2.c: Tratamiento de cadenas.

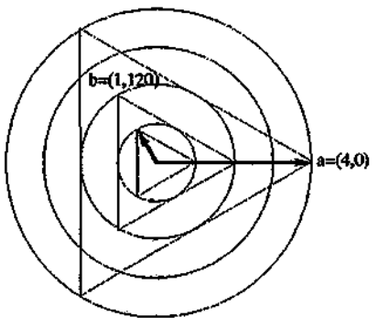
Obsérvese que el papel de los símbolos de tipo s'' se limita a asegurar la interpretación gráfica correcta porque la regla de evolución del sistema, derivación tras derivación, se obtiene exclusivamente gracias a los símbolos de tipo s'.

Cada símbolo del alfabeto de partida (s) tiene, por tanto, asociado un símbolo (s') o dos símbolos (s' y s'') en el nuevo alfabeto. Para cada regla que tenga como parte izquierda el símbolo s generamos una regla (con s' en la parte izquierda y la cadena derivada de la parte derecha de partida como parte derecha) en el conjunto de reglas de producción objetivo (P') y otra (si el vector asociado tiene módulo mayor que r) con s'' en la parte izquierda y que genera la cadena vacía ($s'' ::= \lambda$).

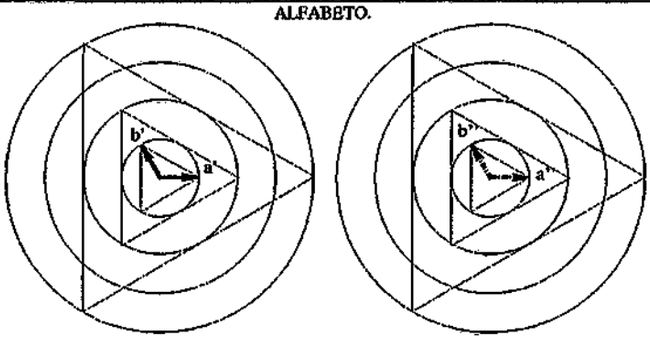
La figura 5.t2.d muestra un ejemplo del tratamiento de cadenas para reglas y axioma. Las sucesivas derivaciones de la cadena "a" generan una escalera. En la parte izquierda de la figura se muestra la interpretación gráfica del alfabeto, las reglas y las tres primeras derivaciones del sistema. En la figura de la derecha se muestra un sistema tortuga gráficamente equivalente, obtenido según las observaciones anteriores. Los segmentos discontinuos representan los símbolos que desaparecen de derivación en derivación por tener reglas del tipo $s'' ::= \lambda$, que se han omitido en la figura.

Para la representación de las reglas se ha dibujado la interpretación gráfica de las partes izquierdas y derecha.

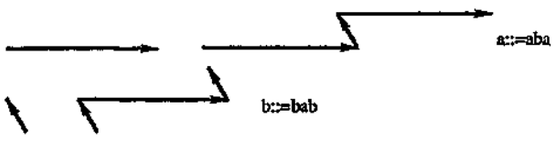
ALFABETO.



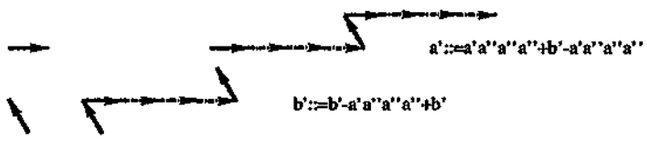
ALFABETO.



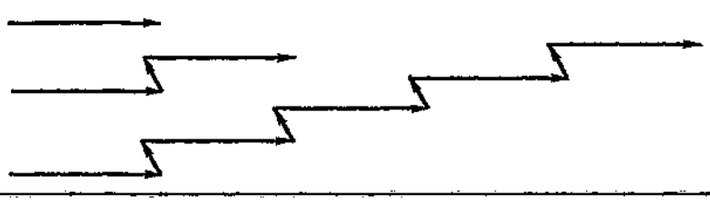
REGLAS.



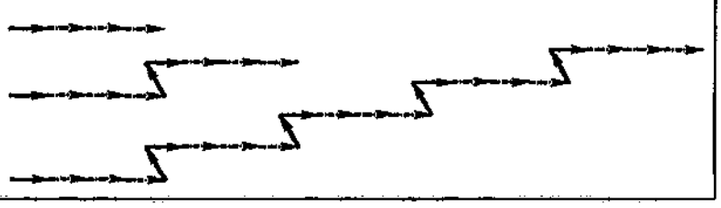
REGLAS.



TRES PRIMERAS DERIVACIONES A PARTIR DE a.



TRES PRIMERAS DERIVACIONES A PARTIR DE a'a''a''a''=C[a,0].



Cuando todos los vectores tienen el mismo módulo sólo se necesita la familia de símbolos s'. Respecto a las cadenas de símbolos + y − necesarios para alcanzar la inclinación de los vectores originales, es necesario no olvidar que cuando se está interpretando una cadena (o una subcadena) en la interpretación gráfica de tortuga, ésta tiene en cada momento una inclinación no necesariamente nula. Es necesario recordar la inclinación de la tortuga al principio de la cadena, porque los incrementos y decrementos necesarios para alcanzar el ángulo deseado son relativos a la misma.

La figura 5.t2.e muestra esta circunstancia.

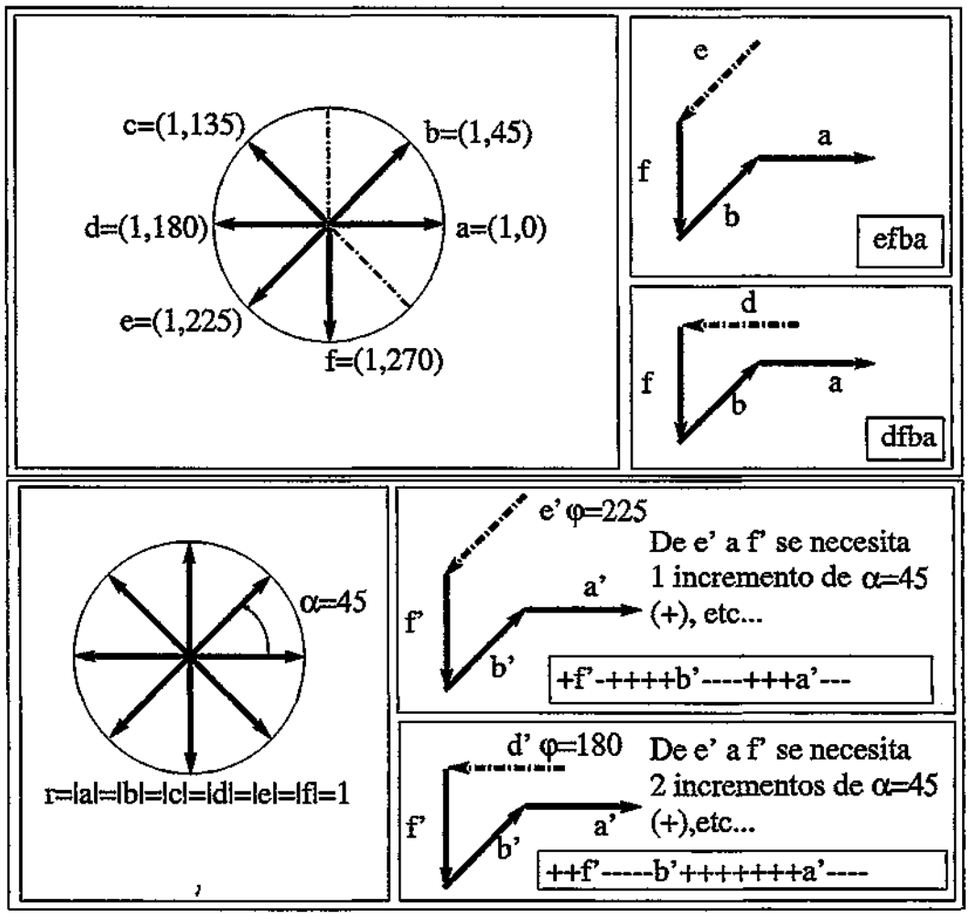


Figura 5.t2.e: Incrementos relativos a la inclinación inicial.

En ella puede observarse que la subcadena de partida ("*fba*") genera distintas subcadenas con interpretación gráfica tortuga, dependiendo de la inclinación con que se comienza a transformar. En el primer caso, el símbolo *e* tiene asociado un símbolo (e') con inclinación 225 grados. Esto hace que para alcanzar la inclinación del símbolo f' (270 grados), en el que se transforma el primero de la subcadena (f), es necesario un incremento de α (una subcadena "+"). Cada símbolo debe dejar la inclinación de la tortuga inalterada, por lo que tras el símbolo f' se

necesita una subcadena opuesta en cuanto incrementos de α. Esta cadena es $-$. En la segunda situación la cabeza de la tortuga tiene la inclinación del símbolo d', que es 180. El incremento necesario para llegar a 270 ($\varphi(f')$) es ahora de dos. La subcadena necesaria es ahora "++". La subcadena que seguirá al símbolo f' es, por tanto, "−−". El mismo fenómeno se observa en los demás simbolos de la cadena.

Obsérvese que siempre es posible expresar el incremento necesario para llegar desde φ a la inclinación de cualquier símbolo de al menos dos maneras distintas: mediante una cadena de q símbolos $+$ o mediante una cadena de $n-q$ símbolos $-$. De hecho, éstas son las cadenas más cortas compuestas exclusivamente por símbolos $+$ o por símbolos $-$ que lo consiguen.

En el algoritmo que constituirá la prueba de este resultado se utilizará la cadena más corta posible.

7.5.2 Demostración formal

La demostración del teorema 2 consiste en la descripción de un algoritmo que obtiene el sistema (esquema) mencionado a partir del de partida. A lo largo de la demostración se utilizará la siguiente notación:

- $x(s)$ para referirse a la parte derecha de una regla que tenga al símbolo s como parte izquierda ($s ::= x(s)$).
- $m(s)$ y $\phi(s)$ para referirse respectivamente al módulo y el ángulo del símbolo s, como se definió en el capítulo 6.
- $M(G_V)$ y $\Theta(G_V)$ para referirse respectivamente al conjunto de módulos y ángulos asociados a Σ, como se definió en la introducción.
- El punto (.) representa la concatenación de cadenas, como se definió en la sección 2.1.2.

Sea $L = (\Sigma, P, w)$ el sistema RRVGD0L de partida con una interpretación gráfica vectorial $G_V : \Sigma \to \{0,1\} \times \Re^2$. Se llevan a cabo las siguientes operaciones:

- Obtención de los conjuntos $\Theta(G_V)$ y $M(G_V)$.
- Al ser un sistema RRVGD0L y al cumplirse $\Theta(G_V) \subset \Re \wedge M(G_V) \subset \Re$, puede afirmarse la existencia de α y r tales que todos los módulos y todos los ángulos de los conjuntos $\Theta(G_V)$ y $M(G_V)$ son respectivamente múltiplos enteros positivos de α y r . El siguiente paso es calcular esos valores r y α.
- A continuación se construye el sistema D0L $L' = (\Sigma', P', w')$ de forma que:
  - Construcción de Σ':

$$\forall s \in \Sigma \Rightarrow s', s'' \in \Sigma'$$
$$\{+, -\} \subset \Sigma'$$

  - Construcción de P':

$$\forall s ::= x(s) \in P \Rightarrow \begin{cases} s' ::= C[x(s), \phi(s)] \in P' \\ s'' ::= \lambda \in P' \end{cases}$$
$$\{+ :: +, - ::= -\} \subset P'$$

- Construcción de $C[x,p] : \Sigma^* \times \Re \rightarrow \Sigma'^*$ sirve para transformar cadenas de símbolos del alfabeto Σ (supuesta una inclinación inicial p) en cadenas de símbolos del alfabeto Σ'^* y se define recursivamente así::
  - $\forall p \in \Re \Rightarrow C[\lambda, p] = \lambda$
  - $\forall p \in \Re, \forall s \in \Sigma | G_V(s) \neq (0,0,0) \Rightarrow C[s,p] = A.B.C$,,

$$\begin{cases} A = \begin{cases} +...+,, |A| = \frac{(\phi(s)-p)}{\alpha} & si \quad \phi(s) \geq p \\ -...-,, |A| = \frac{(p-\phi(s))}{\alpha} & si \quad \phi(s) < p \end{cases} \\ B = s's''...s'',, |B| = \frac{m(s)}{r} \\ C = \begin{cases} -...-,, |A| = \frac{(\phi(s)-p)}{\alpha} & si \quad \phi(s) \geq p \\ +...+,, |A| = \frac{(p-\phi(s))}{\alpha} & si \quad \phi(s) < p \end{cases} \end{cases}$$

  - $\forall p \in \Re, \forall s \in \Sigma | G_V(s) = (0,0,0) \Rightarrow C[s,p] = s'$
  - $\forall p \in \Re, \forall s \in \Sigma, \forall y \in \Sigma^* \Rightarrow C[s.y,p] = C[s,p].C[y,p]$
- Construcción de la interpretación gráfica de tortuga.
  - $G_T = (D, M, U, \alpha, r)$, donde
    - $D = \{s' | G_V(s) = (1, v_x, v_y) \wedge v_x \neq 0 \wedge v_y \neq 0\}$
    - $M = \{s' | G_V(s) = (0, v_x, v_y) \wedge v_x \neq 0 \wedge v_y \neq 0\}$
    - $U = \{s' | G_V(s) = (0,0,0)\}$
    - α y r fueron calculados en el segundo paso.
- Construcción del axioma.
  - $w' = C[w, 0]$.

Por construcción, la curva representada por el gráfico de Lindenmayer (L, G_V) es la misma que la representada por (L', G_T) . Por tanto, L y L' son gráficamente equivalentes.

La segunda parte del teorema, que se refiere a esquemas de Lindenmayer, resulta trivial. Dado un esquema de Lindenmayer con interpretación vectorial el algoritmo es capaz de construir un esquema de Lindenmayer asociado. Para cada axioma que se utilice con el primer sistema, el algoritmo construye un axioma en el segundo, de tal modo que los sistemas L obtenidos son gráficamente equivalentes. Esto demuestra el resultado.

7.6 Ejemplo de aplicación del teorema 2.

Se tomará como gráfico de Lindenmayer de partida el que se obtuvo como ejemplo de aplicación del teorema 1. Se ha renombrado los conjuntos $(L, \Sigma, P$ en lugar de $L', \Sigma', P')$ para que se adecúen a la notación del algoritmo.

$$
\left(L = \left(\begin{array}{c} \Sigma = \{A, B, C, D, E, F\}, \\ P = \left\{ \begin{array}{l} A ::= AFBA, \\ B ::= BACB, \\ C ::= CBDC, \\ D ::= DCED, \\ E ::= EDFE, \\ F ::= FEAF \end{array} \right\}, \\ w = ACE \end{array} \right), \right.
$$

$$
\left. G_V(s) = \left\{ \begin{array}{lll} (1,1,0) & si & s = A \\ \left(1, 0.5, \frac{\sqrt{3}}{2}\right) & si & s = B \\ \left(1, -0.5, \frac{\sqrt{3}}{2}\right) & si & s = C \\ (1,-1,0) & si & s = D \\ \left(1, -0.5, -\frac{\sqrt{3}}{2}\right) & si & s = E \\ \left(1, 0.5, -\frac{\sqrt{3}}{2}\right) & si & s = F \end{array} \right\} \right)
$$

A partir de este sistema, y mediante la aplicación del algorimo del teorema 2, puede obtenerse otro sistema con interpretación de tortuga que también representa la curva de copo de nieve de Koch.

Puede observarse por simple inspección de G_V que los valores de r y α son respectivamente 1 y 60 grados ($\frac{\pi}{3}$ radianes).

Al aplicar el algoritmo del teorema 2, se obtiene el siguiente gráfico de Lindenmayer.

$$
\left(L' = \left(\begin{array}{c} \Sigma' = \{A', B', C', D', E', F'\}, \\ P' = \left\{ \begin{array}{l} A' ::= A' - F' + + B' - A', \\ B' ::= B' - A' + + C' - B', \\ C' ::= C' - B' + + D' - C', \\ D' ::= D' - C' + + E' - D', \\ E' ::= E' - D' + + F' - E', \\ F' ::= F' - E' + + A' - F', \\ + ::= + \\ - ::= - \end{array} \right\}, \\ w' = A' + + C' - - - - E' + + \end{array} \right), \right.
$$

$$
\left. G_V(s) = \left\{ \begin{array}{lll} (1,1,0) & si & s = A \\ \left(1, 0.5, \frac{\sqrt{3}}{2}\right) & si & s = B \\ \left(1, -0.5, \frac{\sqrt{3}}{2}\right) & si & s = C \\ (1,-1,0) & si & s = D \\ \left(1, -0.5, -\frac{\sqrt{3}}{2}\right) & si & s = E \\ \left(1, 0.5, -\frac{\sqrt{3}}{2}\right) & si & s = F \end{array} \right\} \right)
$$

Es fácil comprobar que el fractal representado por este gráfico de Lindenmayer es la curva de copo de nieve de Koch de la figura 5.t1.i de la sección 7.3.3.

Capítulo 8

Cálculo de la dimensión de algunos fractales

8.1 Ejemplo de aplicación de la dimensión de Richardson-Mandelbrot

En el capítulo 5 se definió la dimensión de Richardson-Mandelbrot.

Esta forma de calcular la dimensión fractal de una curva se basa en la aplicación de un patrón o regla de longitud fija y el estudio de la proporción entre el número de veces que es necesario aplicarlo para recorrer la curva (número de pasos de esa longitud necesarios para recorrer la curva) y la longitud del patrón. El proceso es análogo a separar las patas de un compás la distancia seleccionada y aplicarlo sobre la curva tantos pasos como se necesite para llegar al final. Se anota la longitud y el número de pasos y se reduce la longitud que separa las patas repitiendo el proceso hasta que la proporción se estabilice (límite). La dimensión fractal, según este método, está asociada al límite de esa proporción cuando la longitud del patrón disminuye.

En las figuras desde la 6.c.0.f hasta la 6.c.0.h se muestra un ejemplo de aplicación de este método a una sección de la curva de copo de nieve de von Koch. Se comienza con una longitud de patrón que abarca la curva completa y que se puede considerar la unidad. Es suficiente con aplicar una vez el patrón para recorrer la curva completa. En el siguiente paso se considera una longitud del patrón de $\frac{1}{3}$. En este caso son necesarios 4 desplazamientos. En la última se toma como longitud del patrón $\frac{1}{9}$, y son necesarios 16 desplazamientos. En las figuras puede observarse como, efectivamente, las intrincadas ramificaciones de la curva no son apreciables con un detalle mayor que el valor p_l utilizado: es decir, la parte de la gráfica dentro del alcance del patrón es reemplazada por un segmento de su longitud. Para el cálculo de la dimensión, por tanto, la curva real aporta la misma información que el recorrido sobre ella marcado con flechas.

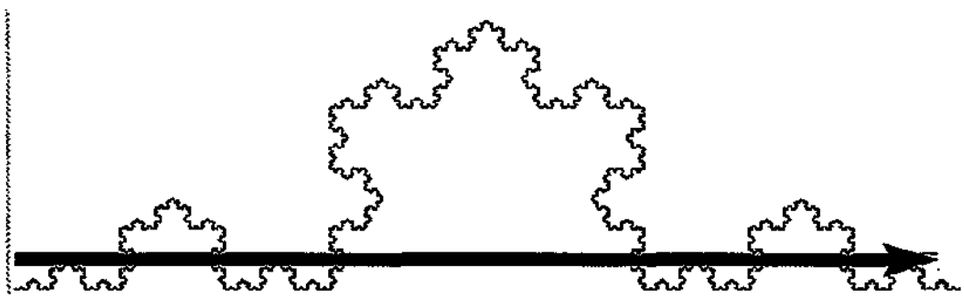


Figura 6.c.0.f: Dimensión de copo de nieve de von Koch, $p_l = 1$, $N(p_l) = 1$.

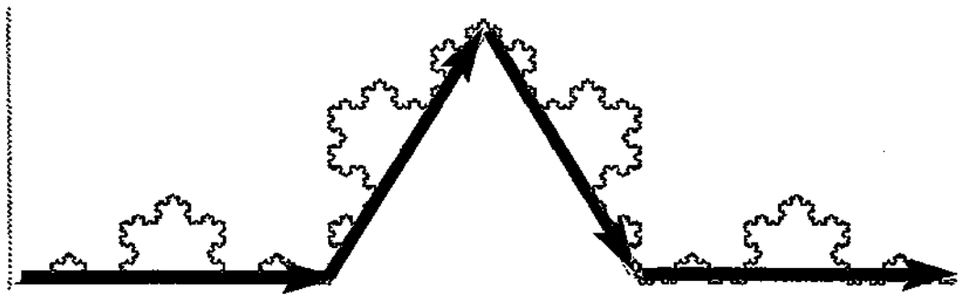


Figura 6.c.0.g: Dimensión de copo de nieve de von Koch, $p_l = \frac{1}{3}$, $N(p_l) = 4$.

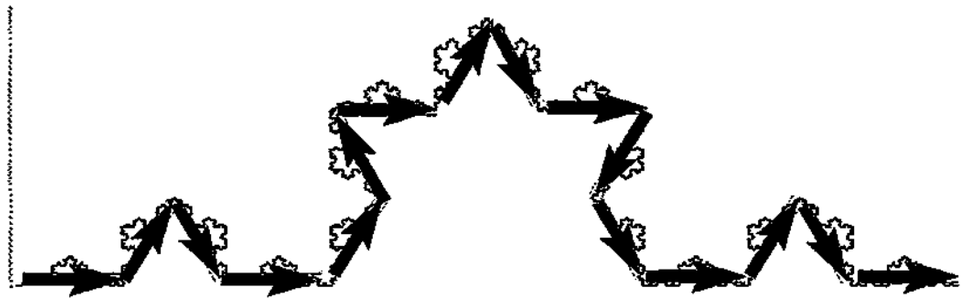


Figura 6.c.0.h: Dimensión de copo de nieve de von Koch, $p_l = \frac{1}{9}$, $N(p_l) = 16$.

El límite converge rápidamente, como se muestra a continuación:

- $p_l = 1$, el cociente del límite es $\frac{-\log(1)}{\log(1)}$.
- $p_l = \frac{1}{3}$, el cociente del límite es $\frac{-\log(4)}{\log\left(\frac{1}{3}\right)} = \frac{-\log(4)}{\log(3^{-1})} = \frac{\log(4)}{\log(3)}$.
- $p_l = \frac{1}{9}$, el cociente del límite es $\frac{-\log(16)}{\log\left(\frac{1}{9}\right)} = \frac{-\log\left(4^2\right)}{\log(3^{-2})} = \frac{-2\log(4)}{2\log(3)} = \frac{\log(4)}{\log(3)}$.
- Desde aquí el cociente no varía por lo que el límite toma este valor $D_p = \frac{\log(4)}{\log(3)}$.

8.2 Cálculo de la dimensión de algunos fractales mediante el estudio de gráficos tortuga

8.2.1 Observaciones previas

Al analizar la curva formada por la concatenación de las flechas de longitud p_l en las figuras anteriores puede apreciarse la analogía con la interpretación gráfica de las sucesivas derivaciones del axioma del siguiente gráfico tortuga que, como se mostró en la sección 7.3.3, representa el fractal de copo de nieve de von Koch.

$$G_K = \left(L = \left(P = \left\{ \begin{array}{c} \Sigma = \{F, +, -\}, \\ F ::= F - F + +F - F, \\ + ::= +, \\ - ::= - \end{array} \right\}, \begin{array}{c} \\ \\ \\ w = F \end{array} \right) \right.,$$

$$\left. G_T = (D = \{F\}, M = \Phi, U = \Phi, 60) \right)$$

La curva estudiada es realmente la parte superior del copo de nieve, por lo que es suficiente utilizar como axioma la cadena "F".

En las figuras desde la 6.c.0.i hasta la 6.c.0.k se muestra las interpretaciones gráficas de las tres primeras derivaciones.

Figura 6.c.0.i: Interpretación gráfica de G_K.

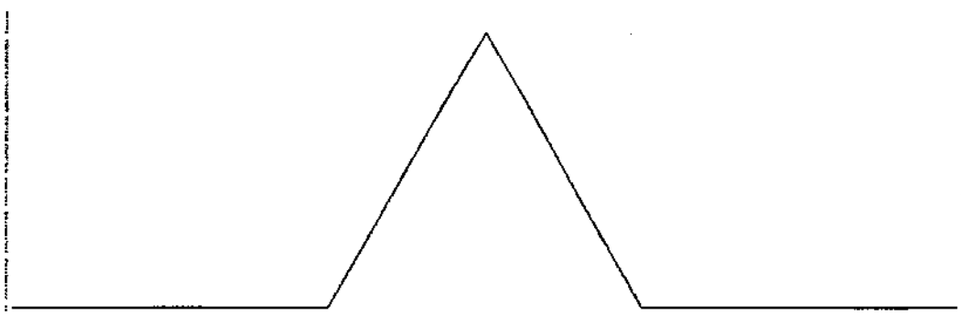


Figura 6.c.0.j: Interpretación gráfica de G_K.

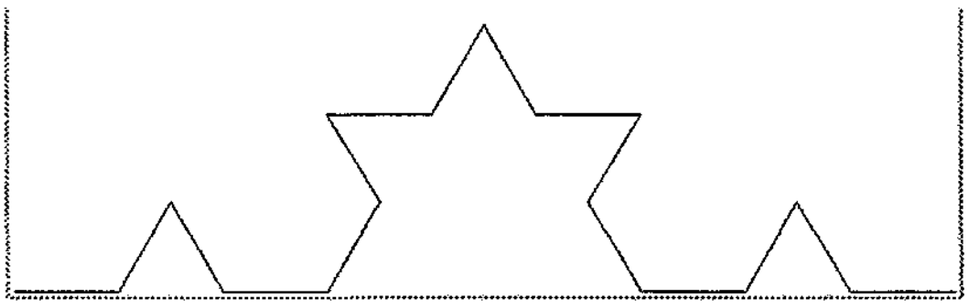


Figura 6.c.0.k: Interpretación gráfica de G_K.

Puede observarse también que los argumentos de las funciones *logaritmo* del numerador y denominador son respectivamente el número de trazos de la representación gráfica de la cadena (número de símbolos visibles de la misma, es decir, que pertenecen a D) y la longitud de la misma (medida en la dirección del segmento inicial y tomando como unidad p_l).

En las figuras desde la 6.c.0.l hasta la 6.c.0.n se muestra con más claridad estas cantidades. La parte superior de cada gráfico cuenta el número de trazos dibujables de la palabra. La inferior cuenta el número de tramos iguales al desplazamiento mínimo del gráfico tortuga en la dirección de la interpretación gráfica de partida.

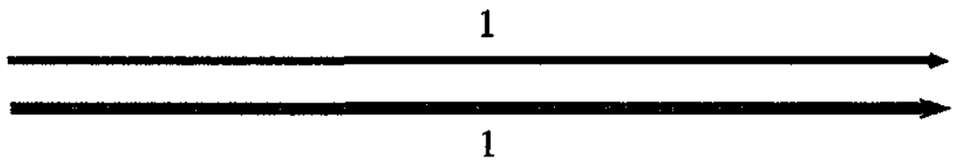


Figura 6.c.0.l: Cantidades significativas para la dimensión de G_K.

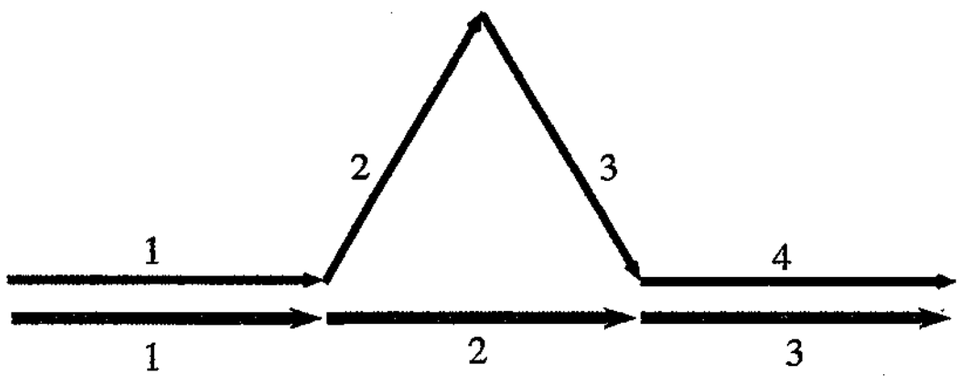


Figura 6.c.0.m: Cantidades significativas para la dimensión de G_K.

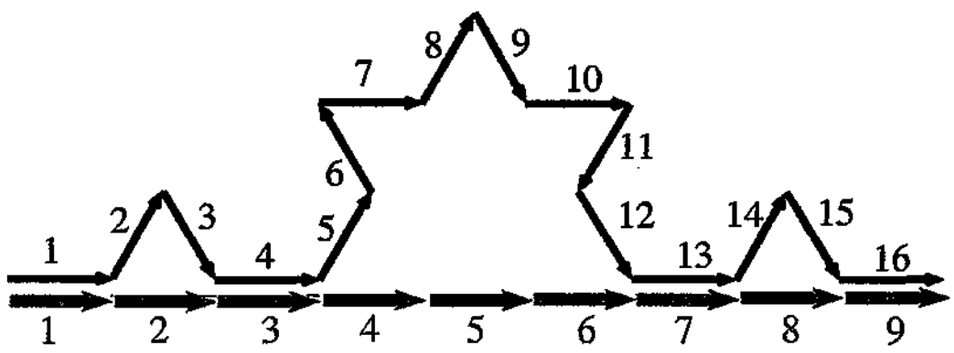


Figura 6.c.0.n: Cantidades significativas para la dimensión de G_K.

Este gráfico Lindenmayer es muy sencillo. La estructura de sus derivaciones corresponde a una sola regla. Cabe preguntar qué ocurriría con sistemas más complejos y cómo afectaría la mezcla de reglas con estructuras distintas en el cálculo de la dimensión fractal.

Estas reflexiones sugieren la definición de una nueva dimensión fractal que se presentará tras unos conceptos previos.

8.2.2 Definición

Se llama *gráfico de Lindenmayer con estructura simple* a todo gráfico de Lindenmayer cuyo sistema cumple la condición de que cada una de sus reglas de producción propagativas (cuya parte derecha no sea λ), cuya parte izquierda no se uno de los símbolos triviales (+, −, (,)), cumple la siguiente condición: la parte derecha de la regla contiene el mismo número de símbolos dibujables (elementos de D), el mismo número de símbolos de movimiento (elementos de M) y el mismo número de símbolos no gráficos distintos de + y −.

Formalmente

$GL = (L = (\Sigma, P, w), G_T = (D, M, U, \alpha, m))$ gráfico tortuga de Lindenmayer tiene estructura simple $\Leftrightarrow \exists t_D, t_M, t_U \in \aleph |\ \ \forall X ::= y \in P, , X \in \Sigma - \{+, -\} \Rightarrow$

$$\begin{cases} \#\{A : A \in y \wedge A \in D\} = t_D \\ \#\{A : A \in y \wedge A \in M\} = t_M \\ \#\{A : A \in y \wedge A \in U - \{+, -\}\} = t_U \end{cases}$$

8.2.3 Ejemplos: otras representaciones de la curva de copo de nieve de von Koch

El siguiente gráfico de Lindenmayer también representa la curva de copo de nieve de von Koch mostrada en la figura 5.t.1.i.

$$G_1 = \left(L_1 = \left(\begin{array}{c} \Sigma_1 = \{F, G, +, -\}, \\ P_1 = \left\{ \begin{array}{l} F ::= -G + F + G-, \\ G ::= +F - G - F+, \\ + ::= +, \\ - ::= - \end{array} \right\}, \\ w_1 = F \end{array} \right) \right)$$

$$G_{L_1} = \left(\begin{array}{l} D_1 = \{F, G\} \\ U_1 = \{+, -\} \\ M_1 = \Phi \\ \alpha_1 = 60^o \\ m_1 = 1 \end{array} \right)$$

Este gráfico de Lindenmayer tiene estructura simple con $t_D = 3$ y $t_M = t_U = 0$, como se puede comprobar fácilmente tras el estudio de las cadenas $-G+F+G-$ y $+F-G-F+$, que son las partes derechas de las reglas cuyas partes izquierdas no son símbolos triviales.

El siguiente gráfico de Lindenmayer también representa la curva de copo de nieve de von Koch.

$$G_3 = \left(L_3 = \left(\begin{array}{c} \Sigma_3 = \{F, P, Q, R, S, T, U, +, -\}, \\ P_3 = \left\{ \begin{array}{l} P ::= PFU - FQ + F - PF, \\ Q ::= Q + F - PFR + +F - -Q + F-, \\ R ::= R + +F - -Q + F, \\ -S + + + F - - - R + +F - -, \\ S ::= S + + + F - - - R + +F \\ - - T - -F + +S + + + F - - -, \\ T ::= T - -F + +S + + + F \\ - - -U - F + T - -F + +, \\ U ::= U - F + T - -F + +PFU - F+, \\ F ::= \lambda, \\ + ::= +, \\ - ::= - \end{array} \right\}, \\ w_3 = P + +P + +P \end{array} \right) \right)$$

$$G_{L_3} = \left(\begin{array}{l} D_3 = \{F,\} \\ U_3 = \{+, -\} \\ M_3 = \{P, Q, R, S, T, U\} \\ \alpha_3 = 60^o \\ m_3 = 1 \end{array} \right)$$

En este caso la estructura es simple con $t_D = 4$, $t_M = 4$ y $t_U = 0$.

8.2.4 Definición

Se llama *dimensión de Lindenmayer* de una regla r de un gráfico tortuga de Lindenmayer con estructura simple, y se escribe $D_L(r)$ al siguiente límite:

$$D_L(r) = \lim_{n_{dr} \to \infty} \left(\frac{\log(L^C_{n_{dr}})}{\log(L^D_{n_{dr}})} \right)$$

Donde

- n_{dr} es el número de derivaciones a los que se somete la parte izquierda de la regla r.
- $L^D_{n_{dr}}$ es la longitud abarcada por la interpretación gráfica de la curva en la dirección

de la interpretación de la primera derivación y tomando como unidad la longitud del desplazamiento mínimo de la interpretación gráfica.

- L^C_{ndr} es la longitud recorrida por la interpretación gráfica de la cadena.

8.2.5 Consecuencia

En todo gráfico de Lindenmayer con estructura simple, todas las reglas con parte izquierda no trivial tienen la misma dimensión de Lindenmayer.

8.2.6 Definición

Se llama *dimensión de Lindenmayer de un gráfico tortuga de Lindenmayer con estructura simple,* y se escribe D_L a la dimensión de Lindenmayer de cualquiera de las reglas de su sistema L que tenga como parte izquierda un elemento de $\Sigma - \{+, -\}$.

8.2.7 Dificultades de la definición propuesta

La precisión

La definición de esta dimensión calcula un límite cuando el número de derivaciones tiende a infinito ya que el fractal es el límite de la curva cuando el número de derivaciones se hace infinitamente grande. Cualquier implementación en un computador de un algoritmo que calcule esta dimensión se verá obligado a manipular cantidades como los módulos de los vectores que componen los caminos, las coordenadas de los extremos, etc... Estos números son en general inconmensurables por su dependencia trigonométrica con los ángulos involucrados en el proceso. Los ordenadores manipulan aproximaciones racionales de los números irracionales. El correcto comportamiento de los programas se consigue gracias a que se puede utilizar la precisión adecuada para que el error de la aproximación sea despreciable. El cálculo del límite cuando el número de derivaciones se hace infinito hace que los desplazamientos tengan un módulo que tiende a 0. Es decir, a medida que se aproxime al infinito los puntos significativos en el cálculo de la dimensión se harán infinitamente próximos sin que se pueda fijar en general una cota superior para el número de derivaciones o para el error permitido. Por otro lado la naturaleza del límite hace que la propagación de errores sea muy perjudicial, se puede obtener un resultado erróneo para la dimensión por esta causa.

Los solapamientos

Ejemplo 1 El siguiente gráfico de Lindenmayer

$$G_4 = \left(L_4 = \left(\begin{array}{c} \Sigma_4 = \{F, +, -\}, \\ P_4 = \left\{ \begin{array}{l} F ::= F + FF + + + F + + F - \\ FF + + + F + + F - - - F, \\ + ::= +, \\ - ::= - \end{array} \right\}, \\ w_4 = F + + F + + F + + F \end{array} \right), \quad G_{L_4} = \left(\begin{array}{l} D_4 = \{F\} \\ U_4 = \{+, -\} \\ M_4 = \Phi \\ \alpha_4 = 45^o \\ m_4 = 1 \end{array} \right) \right)$$

Muestra una peculiaridad en la interpretación gráfica de la parte derecha de la regla cuya parte izquierda es el símbolo F. Puede observarse en la figura 6.c.0.o que hay un tramo por el que el recorrido de la interpretación gráfica pasa dos veces.

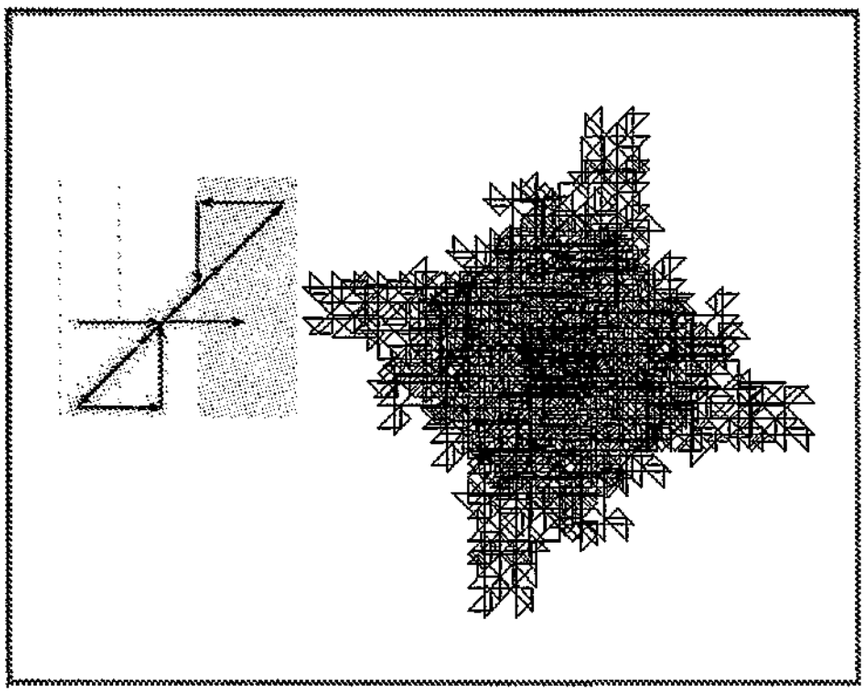


Figura 6.c.0.o: Ejemplo de regla que solapa.

En la parte derecha de la figura se muestra la interpretación gráfica de la palabra obtenida a partir del axioma tras tres derivaciones.

Ejemplo 2 Analícese el siguiente gráfico de Lindenmayer

$$G_5 = \left(\begin{array}{c} L_5 = \left(\begin{array}{c} \Sigma_5 = \{F, +, -\}, \\ P_5 = \left\{ \begin{array}{l} F ::= F + FF - F \\ -FF + F, \\ + ::= +, \\ - ::= - \end{array} \right\}, \\ w_5 = F + F + F + F \end{array} \right) \\ G_{L_5} = \left(\begin{array}{l} D_5 = \{F\} \\ U_5 = \{+, -\} \\ M_5 = \Phi \\ \alpha_5 = 90^\circ \\ m_5 = 1 \end{array} \right) \end{array} \right)$$

Como puede observarse en la figura 6.c.0.p, la interpretación gráfica de la parte derecha de la regla que tiene como parte izquierda el símbolo F no tiene solapamientos.

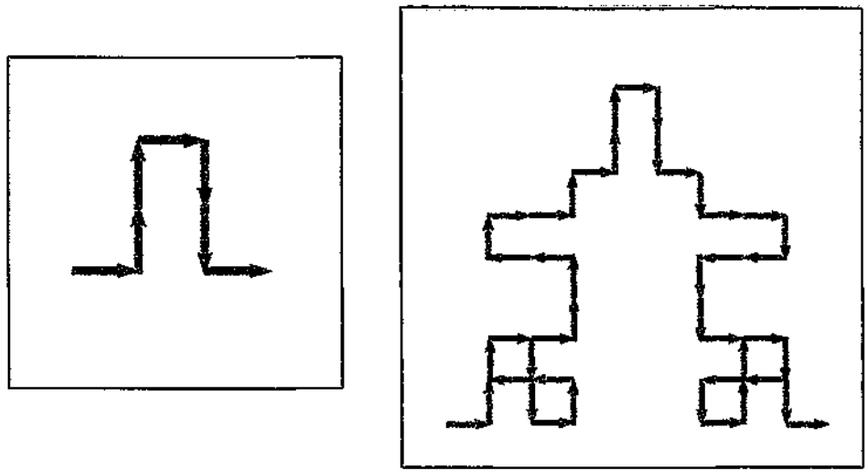


Figura 6.c.0.p: Ejemplo de gráfico Lindenmayer con solapamientos con regla que no solapa.

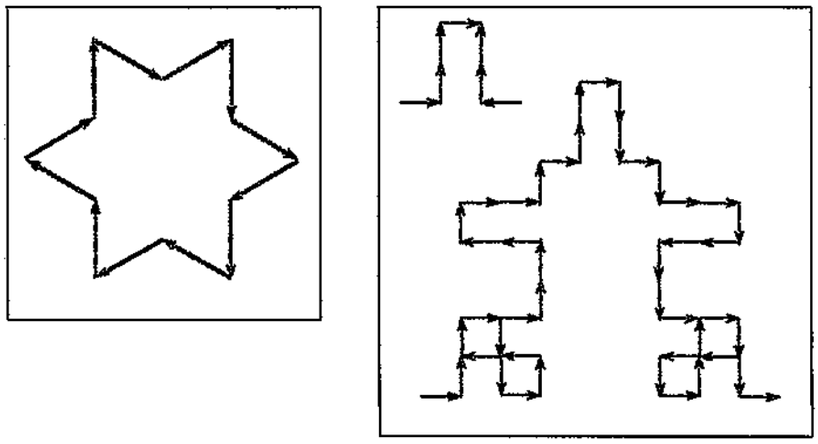
En la parte derecha de la figura se muestra la interpretación gráfica de la segunda derivación del símbolo F. En la parte inferior de ella se han resaltado 4 tramos que son recorridos dos veces.

La cuestión de los solapamientos Dado que la dimensión definida calcula el límite de un cociente basado en dos cantidades, una de las cuales es la longitud recorrida por la curva, en gráficos de Lindenmayer como los de las figuras anteriores se plantean dos alternativas para el cálculo del cociente:

- Calcular la longitud total recorrida por la curva.
- Calcular la longitud "efectiva" recorrida contabilizando sólo una vez los tramos en los que se produzcan solapamientos.

8.2.8 Longitud recorrida

Al interpretar gráficamente cada palabra generada por un sistema L con interpretación gráfica de tortuga, sólo se genera algún trazo para los símbolos dibujables. Esto significa que el módulo de cada trazo multiplicado por el número de símbolos dibujables es igual a la longitud recorrida total, que definimos como la longitud recorrida por la curva que se está representando. En la siguiente gráfica se muestra dos recorridos, el de la izquierda no tiene solapamientos. El de la derecha (cuya figura básica, es decir, la que hay que utilizar para recorrer la figura principal, está dibujada en la esquina superior izquierda, recorre dos veces los tramos resaltados con distinto color.



Se plantea, por tanto, dos alternativas a la hora de calcular el valor de la longitud recorrida:

- Contar repetidamente los recorridos múltiples. En este caso la longitud recorrida es igual a la longitud recorrida total.
- Contar una sola vez los tramos que se solapan: en ese caso, la longitud recorrida efectiva es menor que la longitud recorrida total, que proporciona una cota superior.

A continuación se definirá y propondrá algoritmos para estas dos opciones.

Definición y cálculo de $\ell^{C}_{n_{dr}}$

La *longitud total recorrida por una cadena x de un gráfico tortuga* $G = (L, G_T = (D, M, U, \alpha, m))$, $\ell^{C}_{n_{dr}}(x, G)$, o simplemente $\ell^{C}_{n_{dr}}$ si no hay posible ambigüedad respecto a la cadena y al gráfico tortuga de Lindenmayer considerados, es igual al producto de la longitud elemental (que puede tomarse igual a 1) por el número de símbolos dibujables según G_T de la cadena.

Formalmente

$$\ell^{C}_{n_{dr}}(x, G) = m\#\{A : A \in x \wedge A \in D\}$$

Definición y cálculo de $L^C_{n_{dr}}(x,G)$

La *longitud recorrida efectiva por una cadena x de un gráfico tortuga* $G = (L, G_T = (D, M, U, \alpha, m))$, y se escribirá $L^C_{n_{dr}}(x,G)$ o simplemente $L^C_{n_{dr}}$ si no hay ambigüedad posible respecto a la cadena y al gráfico tortuga de Lindenmayer considerado es igual al valor obtenido como resultado de la ejecución del siguiente algoritmo.

Es necesario utilizar varias funciones:

- *cadena_a_longitud_efectiva(cadena)*, que devuelve un número real que es la longitud efectiva del camino L tortuga asociado a la cadena argumento.
- *segmento_recorrido_a_longitud_efectiva(posición, camino_canónico)*, que devuelve un número real que es la longitud del segmento indicado como una posición dentro del vector de puntos asociado al camino L tortuga de nombre camino_canónico, *dentro* del camino representado precisamente por ese vector de puntos.

El cálculo de $L^C_{n_{dr}}$ se reducirá a llamar a la función *cadena_a_longitud_efectiva* utilizando como argumento la parte derecha de cualquiera de las reglas que tiene en su parte izquierda un símbolo distinto de $-$ y $+$.

A la hora de obtener la longitud efectiva de un segmento (llamaremos desde ahora segmento (p_1,p_2) al segmento cuya longitud efectiva estamos calculando) dentro de un recorrido es necesario tener en cuenta las siguientes consideraciones:

TRATA_SOLAPAMIENTO_COMIENZO:

- Para determinar los solapamientos sólo importa las posiciones relativas de los extremos de los vectores y no su dirección.
- Si se encuentra que el segmento (p_1,p_2) es idéntico a algún otro segmento (p_3,p_4) hay que descartar el segmento (p_1,p_2) porque no aporta longitud al camino. La condición para que los segmentos sean idénticos es que los extremos sean los mismos, es decir $[(p_1 = p_3) \wedge (p_2 = p_4)] \vee [(p_2 = p_3) \wedge (p_1 = p_4)]$. Puesto que se manipula la representación tortuga canónica los puntos serán iguales si y sólo su representación canónica lo es.
- Sólo interesa estudiar solapamientos entre segmentos que tengan la misma pendiente. Si no tienen la misma pendiente los segmentos pertenecerán a rectas que se cortan en un punto y no habrá solapamiento.
- Para determinar que dos vectores del mismo módulo son paralelos es suficiente con comparar que los mismos vectores con punto inicial en el origen sólo se diferencian en su sentido.
- La condición, en función de la representación canónica de los puntos, puede expresarse de la siguiente forma:
  - Trasladar los vectores de forma que empiecen en el origen se consigue restando el origen del extremo.

  - Para que la posible diferencia del sentido de los vectores no afecte se tomará valores absolutos al realizar las diferencias.
  - Por tanto que los vectores sean paralelos es lo mismo que $|p_2 - p_1| = |p_3 - p_4|$

- Una vez que se ha comprobado que los vectores son paralelos, el solapamiento se determinará mediante las posiciones relativas de los extremos de los dos segmentos. El estudio se puede reducir al análisis de las dos siguientes condiciones:

  1. C_1: Los puntos p_1, p_3 y p_4 están alineados y además $p_3 \leq p_1 \leq p_4$ (es decir, $p_1$está entre p_3y p_4)
  2. C_2: Los puntos p_2, p_3 y p_4 están alineados y además $p_3 \leq p_2 \leq p_4$ (es decir, $p_1$está entre p_3y p_4)

- La manera de comprobar la condición $p_j \leq p_i \leq p_k$ (que un punto p_i está alineado con otros dos (p_j y p_k) y además en el segmento formado por ellos) mediante las representaciones canónicas se basa en comprobar que los vectores (p_i, p_j) y (p_i, p_k) tienen la misma dirección y sentido contrario. El estudio se realiza según muestra la siguiente figura: primero se traslada el origen de coordenadas al punto p_i eso posibilita realizar la comprobación de manera más cómoda estudiando el signo de las traslaciones de los puntos p_j y p_k. Estas últimas comprobaciones resultan más sencillas en coordenadas cartesianas.

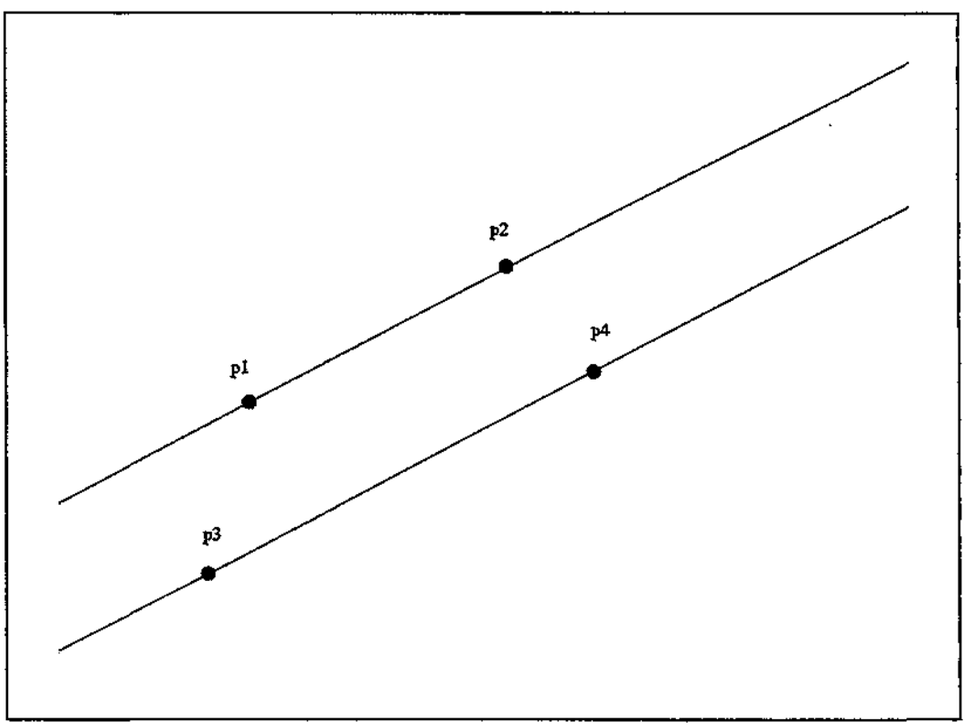
  - Se obtienen los vectores (p_i, p_j) y (p_i, p_k) comenzando en el origen. La representación de esos vectores es, respectivamente, $p_i - p_j$ y $p_i - p_k$.
  - Se pasa a coordenadas cartesianas.
    * Las coordenadas cartesianas de $p_i - p_j$ son (x_1, y_1).
    * Las coordenadas cartesianas de $p_i - p_k$ son (x_2, y_2).
    * La comprobación pasa a ser que los puntos (x_1, y_1), $(0,0)$ y (x_2, y_2) están alineados y que además el signo de las coordenadas de los puntos no nulos es distinto.
      · Puesto que se manipula ahora la recta que pasa por el origen, la condición de alineación de los tres puntos puede expresarse como la igualdad de los productos $x_1 y_2 = x_2 y_1$.
      · La condición de cambio de signo de las coordenadas puede expresarse de la siguiente manera: $x_1 x_2 \leq 0$ y $y_1 y_2 \leq 0$.

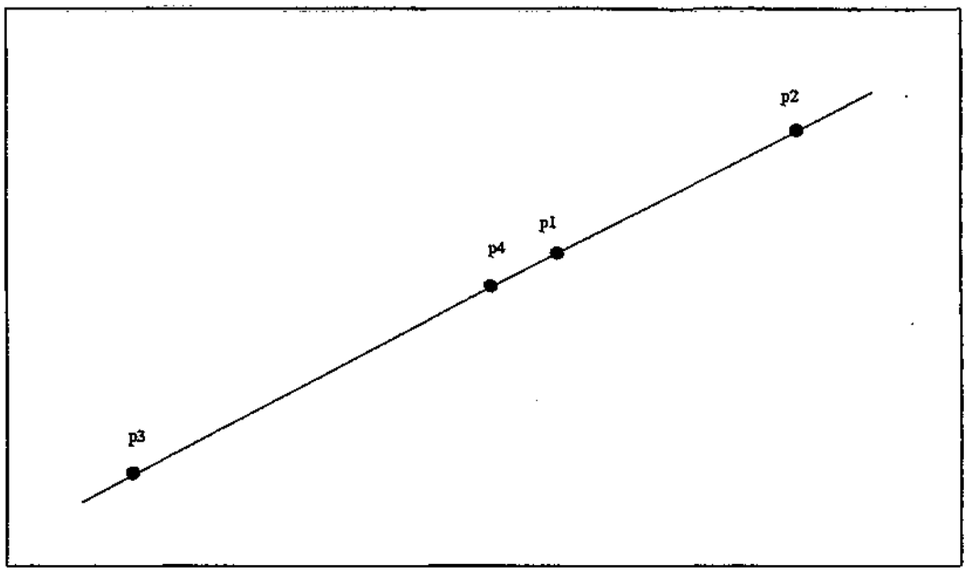
- Se debe estudiar, entonces los siguientes posibles casos:

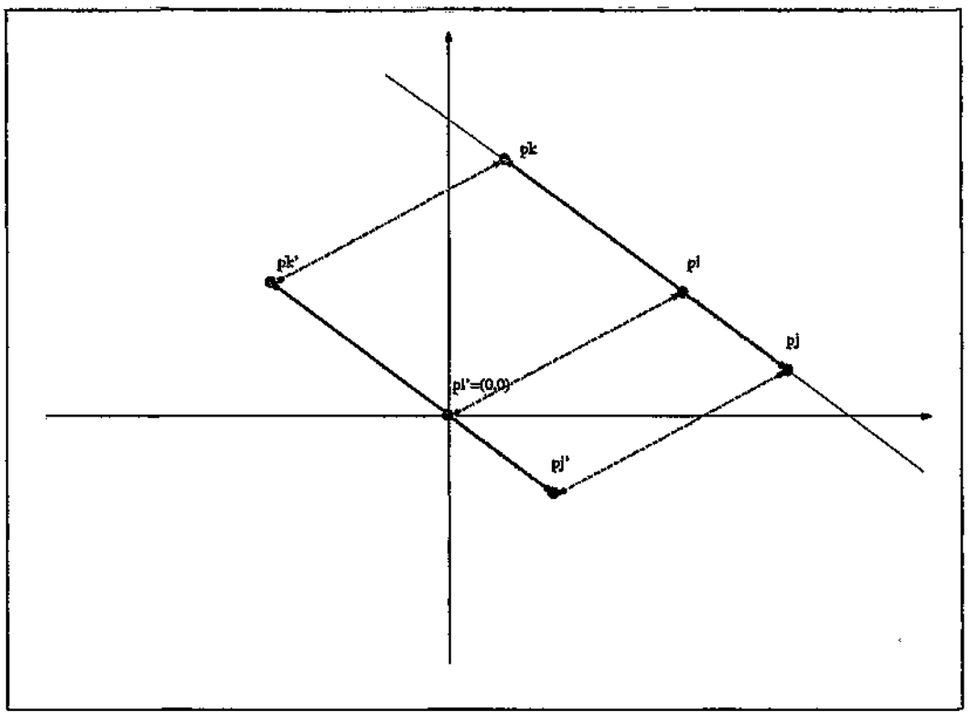
  - $\neg C_1 \wedge \neg C_2$, es decir:
    * (a) O bien no solapan los segmentos, ya sea porque no están alineados o porque no tienen ningún tramo común. Esas dos circunstancias se muestran en las siguientes gráficas.
    * (b) O bien están alineados pero no solapan.
  - $C_1 \wedge C_2$, es decir, el nuevo segmento está totalmente incluido en el viejo, en este caso se termina el tratatamiento y se devuelve 0, el nuevo segmento no aporta longitud efectica al recorrido.

- $C_1 \vee C_2$ (ahora ocurre que sólo una de las dos es cierta, es un or exclusivo)
  - $C_1 \wedge \neg C_2$, es necesario determinar si es p_3 el que está entre p_1 y p_2 o si es p_4 porque sabemos que una de las dos condiciones se tiene que cumplir.
    - Si $p_1 \leq p_3 \leq p_2$, entonces es necesario sustituir en el segmento que está siendo estudiado p_1 por p_3.
    - En otro caso es necesario cambiarlo por p_4.
  - $\neg C_1 \wedge C_2$, es necesario determinar si es p_3 el que está entre p_1 y p_2 o si es p_4 porque sabemos que una de las dos condiciones se tiene que cumplir.
    - Si $p_1 \leq p_3 \leq p_2$, entonces es necesario sustituir en el segmento que está siendo estudiado p_2 por p_3.
    - En otro caso es necesario cambiarlo por p_4.
  - En cualquiera de los dos casos anteriores se continua el estudio con los demás segmentos del camino.

TRATA_SOLAPAMIENTO_FIN:







Se describe a continuación ambas funciones.

cadena_a_longitud_efectiva(cadena) (de tipo real)

- *camino_canónico* ← *cadena_a_representación_canónica(cadena)* Se generar la representación canónica L tortuga de los puntos de la gráfica, se obtiene un vector de rep-

resentaciones canónicas (v) de tal forma que la posición de índice 0 está ocupada por la representación del primero punto. Por tanto *camino_canónico* es un vector de tantas componentes como puntos haya en el camino.

- En este vector de puntos puede encontrarse cada uno de los vectores mediante la identificación de los siguientes pares de puntos:
  - (*camino_canónico*[0], *camino_canónico*[1])
  - (*camino_canónico*[1], *camino_canónico*[2])
  - ...
  - (*camino_canónico*[i], *camino_canónico*[i+1])
  - ...
  - (*camino_canónico*[m-2], *camino_canónico*[m-1])
- Donde m es la dimensión del vector.
- *longitud_efectiva* $\leftarrow$ *0.* (Se inicializa el contador en el que se acumulará la longitud efectiva de la curva a 0).
- Se recorre estos m-1 segmentos realizando para cada uno de ellos el siguiente tratamiento (supongamos un índice i que va de 0 a m-1, por lo tanto el vector que se está estudiando es v[i], v[i+1]):
  - TRATAMIENTO DE SEGMENTO:
  - Si el extremo del vector no es visible el segmento se ignora.
  - SI *camino_canónico*$[i+1][4] = 0$ ENTONCES
    - $i \leftarrow i+1$
    - Ir a TRATAMIENTO DE SEGMENTO:
  - *longitud_efectiva* $\leftarrow$ *longitud_efectiva + segmento_recorrido_a_longitud_efectiva(posición, camino_canónico)*

segmento_recorrido_a_longitud_efectiva(posición, camino_canónico) (de tipo real)

- Se compara el segmento *(camino_canónico[i], camino_canónico[i+1])* con todos los segmentos anteriores obteniendo un segmento "efectivo", ya que puede ser que sea necesario disminuir la longitud del segmento estudiado al eliminar solapamientos.
- $j \leftarrow 0$ (se inicializa un índice que recorrerá el camino canónico desde el primer segmento hasta el segmento estudidado)
- $p_1 \leftarrow$*camino_canónico*$[i]$
- $p_2 \leftarrow$*camino_canónico*$[i+1]$ (se mencionó previamente que el segmento estudiado se llamaría (p_1, p_2))
- MIENTRAS j ¡ i HACER

- $p_3 \leftarrow$*camino_canónico*$[j]$
- $p_4 \leftarrow$*camino_canónico*$[j+1]$ (se mencionó previamente que el segmento del camino con el que se compararía el estudiado se llamaría (p_3, p_4))
- SI $[(p_1 = p_3) \wedge (p_2 = p_4)] \vee [(p_2 = p_3) \wedge (p_1 = p_4)]$ ENTONCES ir a SIGUIENTE SEGMENTO: (el segmento nuevo coincide con alguno de los antiguos)
- Ejecutar las instrucciones contenidas entre las etiquetas TRATA_SOLAPAMIENTO_COMIENZO: y TRATA_SOLAPAMIENTO_FIN:
- SIGUIENTE SEGMENTO: $j \leftarrow j+1$ (se pasa al siguiente segmento del subcamino que lleva del origen al segmento estudiado)

Propiedad

La longitud total recorrida por la representación gráfica es mayor que la efectiva para cualquier curva de cualquier gráfico tortuga de Lindenmayer.

Formalmente

$$\ell^C_{n_{dr}}(x, G) \geq L^C_{n_{dr}}(x, G) \ \forall x, G$$

Definición

Se llama *dimensión total de Lindenmayer de una cadena r de gráfico tortuga de Lindenmayer con estructura simple* y se escribirá D_ℓ a la dimensión de Lindenmayer que utiliza la longitud total recorrida por la curva en el cálculo del límite.

Formalmente

$$D_\ell(r) = \lim_{n_{dr} \to \infty} \left(\frac{\log(\ell^C_{n_{dr}})}{\log(L^D_{n_{dr}})} \right)$$

Definición

Se llama *dimensión total de Lindenmayer de un gráfico tortuga de Lindenmayer con estructura simple* a la dimensión total de cualquiera de sus cadenas (no de borrado) cuya parte izquierda no sea un símbolo trivial (+, −, (,)).

Definición,

Se llama *dimensión efectiva de Lindenmayer* $(D_L(r))$ a la que utilice la longitud efectiva de las curvas en el cálculo del límite:

$$D_L(r) = \lim_{n_{dr} \to \infty} \left(\frac{\log(L^C_{n_{dr}})}{\log(L^D_{n_{dr}})} \right)$$

Determinación de $L^D_{n_{dr}}$

Se está suponiendo a lo largo de este trabajo que la posición inicial es $(0, 0)$ y inclinación inicial es 0^o. Eso significa que la dirección de partida es paralela al eje de abscisas. La longitud recorrida

en esa dirección por una cadena de un gráfico tortuga de Lindenmayer es la distancia del punto origen al punto final.

Formalmente:

- Sea $G = (S, G_T)$ el gráfico tortuga de Lindenmayer considerado.
- Sea $U ::= u$ la regla estudiada.

$$L^D_{n_{dr}} \text{ cumple que } \exists y \in \Re | \quad destino\,(G_T\,(u)) = (x, y) \wedge L^D_{n_{dr}} = \sqrt{x^2 + y^2}$$

Propiedad

La relación numérica entre las dos definiciones de longitud recorrida hacen que se pueda afirmar la siguiente propiedad:

$$D_t(G) \geq D_L(G)\ \forall G$$

Es decir, la dimensión total de Lindenmayer de un gráfico tortuga de Lindenmayer con estructura simple es una cota superior de la dimensión efectiva.

Ejemplos

Tanto la dimensión total como la efectiva de Lindenmayer coinciden con la dimensión de Richardson-Mandelbrot en todos los casos de curvas sin solapamiento analizados. Ejemplos de estos casos son la curva copo de nieve de von Koch y la de Peano cuyas gráficas y dimensiones fueron mostradas en el capítulo 5.

En los ejemplos de curvas con solapamiento presentados en este capítulo las dos dimensiones de Lindenmayer difieren en sus resultados. La dimensión efectiva de Lindenmayer converge más lentamente que la total.

En el caso de la curva de la figura 6.c.0.o, los tres primeros valores obtenidos por el algoritmo de cálculo de la dimensión efectiva descrito son 3.234, 3.276, 3.272 mientras que la dimensión total converge rápidamente a valores cercanos al primero de esta serie.

En el caso de la curva de la figura 6.c.0.p, los tres primeros valores del cálculo de la dimensión efectiva son 1.771, 1.732, 1.708. La dimensión total se comporta de forma similar al caso anterior.

Parte III

Estudio de autómatas celulares mediante sistemas de Lindenmayer

Capítulo 9

Representación de autómatas celulares no probabilistas mediante sistemas L

9.1 Autómatas celulares en dominios de conocimiento distintos de la Informática Teórica

Ya se ha mencionado en la primera sección del capítulo 4 que los autómatas celulares han sido utilizados como una técnica alternativa a la solución de ecuaciones diferenciales para el estudio de sistemas dinámincos complejos.

Es frecuente encontrar ejemplos de autómatas celulares descritos de manera informal e incluso imprecisa. Antes de formalizar con precisión las relaciones entre sistemas L y autómatas celulares se estudiará tres de estos ejemplos con distintas dimensiones entre las que tienen una representación gráfica directa: 1, 2 y 3. Se propondrá una forma de actuar capaz de obtener sistemas L que se comporten igual que los autómatas celulares de partida. Aunque por el momento se omitirá definiciones más precisas, diremos indistintamente que se comportan de la misma manera o que son equivalentes.

9.2 Sistemas L y autómatas celulares

La posibilidad de generar sistemas L equivalentes a autómatas celulares es sugerida por la identificación de algunas similitudes entre ambos sistemas:

- Los dos tienen información inicial que puede ser considerada como su estado de partida.
  - Para un sistema L, esta información incial es el axioma.
  - Para un autómata celular, el conjunto de todos los estados iniciales de sus autómatas finitos, que pueden ser considerados como la configuración inicial del autómata.
- Los dos tienen componentes en los que se codifica la manera de cambiar del sistema.

  - En un sistema L, el conjunto de reglas de producción.
  - En un autómata celular, la función de transición de sus autómatas finitos.

- Las dos arquitecturas generan el próximo estado aplicando su transformación a todos los componentes de la estructura en paralelo.
  - Los sistemas L cambian cada símbolo de la palabra transformada.
  - Los autómatas celulares cambian el estado de cada autómata de la rejilla.

Estas similitudes sugieren la posibilidad de encontrar una convergencia entre autómatas celulares y sistemas L.

El nuestro no es el primer intento en esta dirección. Stephen Wofram ([*Wol*94]) establece una relación entre algunos lenguajes de la jerarquía de Chomsky (regulares y dependientes de contexto) y autómatas celulares unidimensionales. En la referencia [*Koz*93] , se muestra que debe haber una relación estructural entre sistemas L y auómatas celulares porque se les puede aplicar el mismo procedimiento de computación (programación genética). Sin embargo, no realizan ningún esfuerzo para conseguir la conversión entre ambos tipos de sistemas. Los ejemplos de cada arquitectura son, además, diferentes. Tampoco tratan el problema de la equivalencia entre ambos que es el objetivo del presente trabajo.

En referencias más recientes ([*Sip*97], [*Sip*98*a*] y [*Sip*98*b*]), Stauffer y Sipper plantean la cuestión de la equivalencia entre sistemas L y autómatas celulares, aunque en un contexto distinto al nuestro. Por un lado, están interesados fundamentalmente en los autómatas auto replicativos. Por otro lado, convierten sistemas L en autómatas celulares que no son realmente equivalentes a los sistemas L de partida. Se obtiene una equivalencia que depende de la representación gráfica de las cadenas de los sistemas L.

El objetivo de las siguientes secciones es sugerir un procedimiento para generar un sistema L equivalente a un autómata celular predeterminado. Para ello se presenta tres ejemplos. Se ha escogido tres autómatas celulares con diferentes dimensiones: el primero unidimensional, el segundo bidimensional y el tercero tridimensional. Este hecho refuerza la confianza en que el procedimiento pergeñado sea aplicable con suficiente generalidad.

9.3 Autómatas celulares unidimensionales

Como se ha mencionado en el capítulo 4, un autómata celular unidimensional es usualmente concebido como una cadena lineal de autómatas. Una de las vecindades mejor estudiadas es la que incluye, junto al propio autómata, sus dos vecinos más próximos: el vecino a su derecha y el vecino a su izquierda, a los que suele llamarse *predecesor* y *sucesor*, respectivamente.

9.3.1 Autómata celular unidimensional con tres entradas que genera la punta de flecha de Sierpinski

Considérese un conjunto de autómatas finitos dispuestos en fila cuyos estados pertenecen al conjunto $\{0, 1\}$. Esto quiere decir que la rejilla sobre la que los autómatas finitos se disponen

descansa sobre una línea recta. El siguiente estado de cada uno de ellos es función de su propio estado, del de su predecesor y del de su sucesor.

Esta famlia de autómatas puede ser representada de la siguiente manera:

- Se utiliza tres bits para representar los estados de los tres autómatas del vecindario.
- Ya que son tres bits, habrá un total de $2^3 = 8$ posibles configuraciones distintas para el vecindario de estos autómatas.
- Como para cada una de ellas es posible decidir que el autómata estudiado tomará un valor de los dos posibles ($\{0,1\}$) habrá en total $2^{2^3} = 2^8 = 256$ diferentes posibles maneras de especificar el siguiente estado de cada autómata de la rejilla.
- La función de transición suele codificarse mediante el número decimal entre 0 y 255 correspondiente a la transformación.

Por ejemplo, la función correspondiente al código 90 con la notación binaria 01011010 tiene las siguientes salidas.

Estado del predecesor	*Estado del actual*	*Estado del sucesor*	*Siguiente estado actual*
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	0

Es fácil construir un sistema $\langle 1,1\rangle$ DIL cuyas palabras derivadas se corresponden con las generaciones consecutivas de este autómata.

El alfabeto es el conjunto $V_{90} = \{0,1\}$

El conjunto de reglas de producción P_{90} puede obtenerse directamente de las tablas anteriores.

$$P_{90} = \left\{ \begin{array}{ll} 111 ::= 0, & 110 ::= 1, \\ 101 ::= 0, & 100 ::= 1, \\ 011 ::= 1, & 010 ::= 0, \\ 001 ::= 1, & 000 ::= 0 \end{array} \right\}$$

El esquema L de este ejemplo es

$$S_{90} = (V_{90}, P_{90})$$

El axioma es la cadena binaria deducida de la concatenación de estados de la configuración inicial. La figura 7.a muestra las 24 primeras generaciones del autómata celular a partir de una cadena ilimitada con un único autómata con estado 1 y todos los demás a 0. Los estados 1 son representados mediante el símbolo "*". Los estados 0 son representados mediante espacios en blanco.

El axioma para este ejemplo es $\alpha_{90} = 0...010...0$

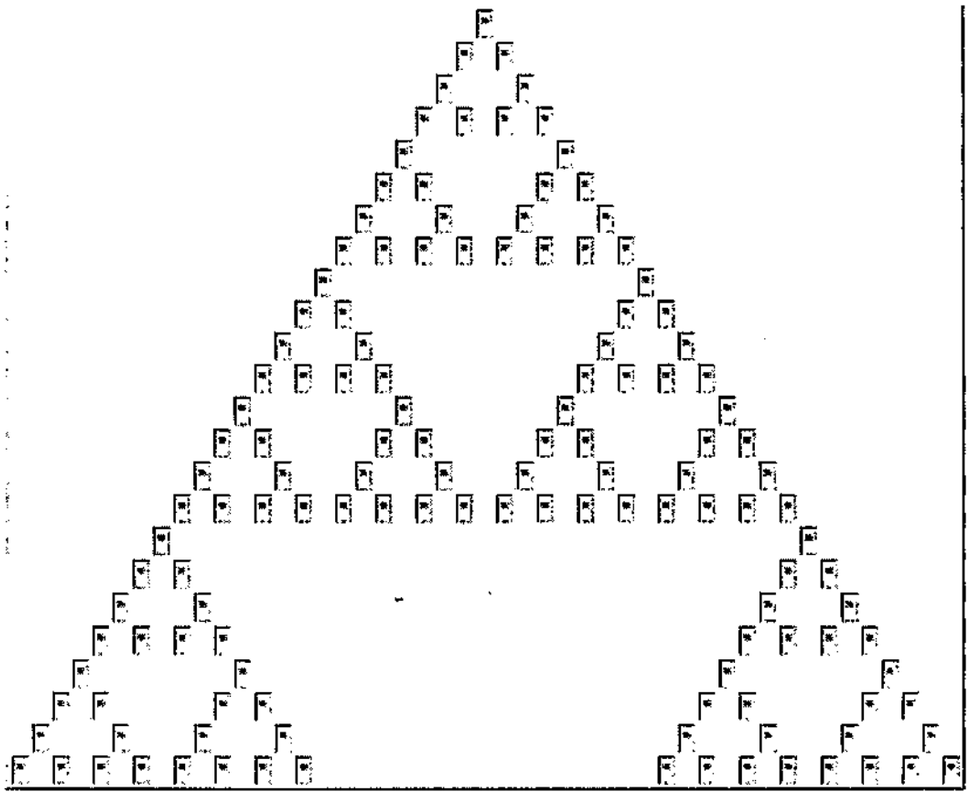


Figura 7.a: Las 24 primeras generaciones del autómata 90 *desde* 0...010...0.

9.4 Autómatas celulares bidimensionales

Como se ha mencionado en el capítulo 4, los autómatas celulares bidimensionales utilizan rejillas bidimensionales. La forma y tamaño de las mismas no están prefijados aunque suelen ser rectangulares e infinitas.

9.4.1 Un autómata celular que simula un ecosistema

Las características del autómata celular que se va a estudiar son las siguientes:

- La rejilla es rectangular y posiblemente infinita.
- Cada celda de la rejilla representa una parcela del territorio donde conviven individuos de diferentes especies por lo que el estado de cada autómata finito de la rejilla corresponde a una combinación de individuos de manera que se cumplen las siguientes condiciones:
  - Hay dos tipos de individuos: presa y depredador.

- Cada celda puede contener un máximo de 4 individuos de tipo presa.
- Hay dos posibles estados para los depredadores, se representarán mediante los símbolos "a" y "b".
- Cada celda puede contener un máximo de 4 individuos depredadores en estado a y un máximo de 4 individuos depredadores en estado b.
- Por consiguiente, el máximo número de individuos permitidos en una celda es igual a 12.

El estado de cada celda es cambiado por dos pasos consecutivos que se alternan a medida que pasa el tiempo.

1. Paso de reproducción y depredación. Las reglas que rigen este paso son aplicadas tantas veces como sea posible y en el orden en el que aparecen. Para ello:
   - Se comprueba la condición para su aplicación en el estado inicial.
   - Se comprueba si existe el espacio necesario para su ejecución en la situación actual en la que se esté, que puede ser un paso intermedio en la reproducción y depredación y, por tanto, distinta a la inicial.
   - A continuación se enumera esas reglas:

   (a) Un depredador en estado "a" *muere* si no hay ninguna presa en la misma celda.

   (b) Un depredador en estado "a" cambia de estado al "b" si hay al menos dos individuos presa en la misma celda y hay suficiente espacio para el nuevo depredador en estado "b" en la celda. Este proceso se lleva a cabo consumiendo un individuo presa que desaparece o *es comido*.

   (c) Un depredador en estado "b" cambia de estado al "a" si no hay ninguna presa en la misma celda.

   (d) Un depredador en estado "b" se transforma en dos depredadores en estado "a" si hay al menos dos individuos presa en la misma celda y hay suficiente sitio para los dos nuevos depredadores en estado "a". Este proceso es la *reproducción de los depredadores* y se realiza con el consumo de una presa que desaparece o *es comida*.

   (e) Las presas se reproducen si hay al menos dos y hay sitio.

2. Paso de movimiento. Este paso considera una vecindad de Von Neumann. Se trata de simular el movimiento no determinista de cada individuo. Se consigue mediante la siguiente regla:
   - Cada individuo puede cambiar de dirección eligiendo aleatoriamente una de las cuatro opciones: norte, sur, este y oeste. Un máximo de un individuo de cada tipo puede ir en la misma dirección.

Se representará ambos pasos mediante dos autómatas celulares distintos que se ejecutarán de forma alternada. Se obtendrá el sistema L equivalente a cada uno de los autómatas de manera

que el sistema L que se comporta como el sistema completo tendrá tablas asociadas a cada uno de los dos autómatas que serán aplicadas con la misma pauta de comportamiento que establece el turno de aplicación de los dos autómatas celulares.

La conducta de este autómata celular muestra muchas de las propiedades de los ecosistemas reales, por ejemplo las oscilaciones de tipo Volterra ([$Alf98$], [$Vol31$]).

Paso de reproducción y depredación

Se utilizará los símbolos "a" y "b" para representar los dos posibles tipos de depredador y la letra "x" para representar una presa. El estado de una celda será representado por una cadena

$$a^{\#a}b^{\#b}x^{\#x}$$

Donde

- $\#a$ es el número de depredadores en el estado a de la celda.
- $\#b$ es el número de depredadores en el estado b de la celda.
- $\#x$ es el número de presas de la celda.
- Un exponente igual a 0 significa que en la celda no hay individuos del tipo asociado al símbolo que tenga ese exponente.
- Cuando sólo hay un individuo de una especie se puede omitir el exponente.

Las reglas de transición que se han descrito previamente pueden ser expresadas utilizando dos cadenas de símbolos, una de ellas para representar las cantidades de individuos necesarias para cada transformación y la otra para representar el número de individuos resultado de la misma. En la siguiente lista se muestran las parejas de cadenas separadas por el símbolo "$\rightarrow$" .

- Regla (a)

$$ax^0 \rightarrow a^0$$

- Regla (b)

$$ax^2 \rightarrow bx$$

- Regla (c)

$$bx^0 \rightarrow a$$

- Regla (d)

$$bx^2 \rightarrow a^2x$$

- Regla (e)

$$x^2 \rightarrow x^3$$

Si, por ejemplo, se estudia el resultado de aplicar estas reglas a un estado con cuatro individuos de cada tipo que se representa mediante la cadena $a^4b^4x^4$, se observa:

- Que las reglas (a) y (c) no son aplicables porque el número de símbolos x es distinto de 0.
- Que la regla (b) se puede aplicar cuatro veces porque hay cuatro individuos de tipo a y suficientes individuos de tipo x. Sin embargo no llega a aplicarse porque no hay sitio suficiente para nuevos individuos de tipo b.
- Lo mismo ocurre con las reglas (d) y (e).

Si, por ejemplo, se estudia un estado con un depredador de tipo a, un depredador de tipo b y tres presas que es representado por la cadena $a^1b^1x^3$, se observa:

- Que las reglas (a) y (c) no son aplicables porque el número de símbolos x es distinto de 0.
- Que la regla (b) puede aplicarse sólo una vez porque hay un único símbolo a. Para que efectivamente se aplique es necesario el suficiente espacio para un nuevo individuo de tipo b. En el momento de aplicarse esta regla (es la primera aplicable) la situación coincide con la de partida. Sólo hay un individuo de tipo b, por lo que hay sitio para tres más. La modificación es posible y por tanto se elimina un individuo de tipo x y el resultado se representa mediante la cadena $a^0b^2x^2$.
- Se ha explicado previamente que para saber si la regla (d) se puede aplicar, hay que consultar el estado de partida y no el resultado de la aplicación de las reglas anteriores. La regla (d) se puede aplicar sólo una vez porque inicialmente sólo hay un individuo de tipo b. Para ser efectivamente aplicada se necesita sitio suficiente para dos nuevos individuos de tipo a. Se ha explicado también que las condiciones relativas al espacio se comprueban en la situación actual que es $a^0b^2x^2$, por tanto no hay ningún individuo de tipo a (habría sitio hasta para cuatro) y la transformación (eliminar un individuo de tipo b y otro de tipo x y crear dos de tipo a) es posible. El resultado se representa mediante la cadena $a^2b^1x^1$.
- La regla (e) es aplicable sólo una vez. En el estado inicial ($a^1b^1x^3$) sólo hay una pareja completa de individuos de tipo x. Se necesitaría espacio para un nuevo individuo de tipo x. En la situación actual ($a^2b^1x^1$) habría sitio hasta para tres más. La transformación es posible y se añade un nuevo individuo de tipo x. El resultado se representa mediante la cadena $a^2b^1x^2$.

Si, por ejemplo, se estudia un estado con tres presas y tres depredadores de tipo a que es representado por la cadena $a^3b^0x^3$, se observa:

- Que las reglas (a) y (c) no son aplicables porque el número de símbolos x es distinto de 0.
- Que la regla (d) no es aplicable porque el número de símbolos b es igual a 0.

- Que la regla (b) es aplicable tres veces porque necesita un individuo de tipo a (y hay tres) y al menos 2 individuos de tipos x (y hay tres). Al ser la primera regla aplicable, las condiciones de espacio disponible también son comprobadas en el estado inicial. Cada aplicación de esta regla necesita sitio para un nuevo individuo de tipo b. Hay sitio para cuatro más. Las tres aplicaciones de la regla son posibles. La primera de ellas transforma un símbolo a en dos símbolos b consumiendo uno x. El resultado se representa mediante la cadena $a^2b^1x^2$. La segunda de ellas hace lo mismo y su resultado se representa mediante la cadena $a^1b^2x^1$. El resultado de la tercera se representa mediante la cadena $a^0b^3x^0$.
- La aplicabilidad de la regla (e) se estudia en la situación de partida que se representa medante la cadena $a^3b^0x^3$. Como sólo hay una pareja completa de símbolos x la regla (e) es aplicable sólo una vez. Se necesita espacio para un nuevo símbolo x. En la situación actual ($a^0b^3x^0$) hay sitio hasta para cuatro. La transformación es posible, se añade un nuevo individuo de tipo x y el resultado se representa mediante la cadena $a^0b^3x^1$.

Sisema L equivalente al autómata del paso de reproducción y depredación

Determinación del alfabeto Es posible construir un sistema L bidimensional cuyas palabras derivadas correspondan a las generaciones sucesivas de este autómata. Para ello se utilizará una rejilla como la del autómata celular pero se asociará un simple símbolo a cada celda en lugar de una palabra completa como se hacía en el autómata. Para ello se asignará un símbolo a cada posible combinación de individuos contenido de una celda. Esta codificación supone la expresión de las reglas con un grado de abstracción más alto: en lugar de representar cada regla describiendo la manera en la que se aplica, la transición de la celda es tratada como un todo y se definirá una regla de producción en el sistema L para expresar de forma explicita cada posible cambio.

Se utilizará para representar el estado de cada celda el símbolo "s" con un subíndice compuesto por la concatenación de los exponentes de la cadena descrita en los párrafos anteiores, ocupando el mismo orden en que aparecen. Por ejemplo:

- El estado descrito por la cadena $a^1b^1x^3$ será representado mediante el símbolo s_{113}.
- El estado descrito por la cadena $a^2b^1x^2$ será representado mediante el símbolo s_{212} .

Y al contrario.

- El símbolo s_{031} representa al estado que tiene como contenido el descrito por la cadena $a^0b^3x^1$.
- Los símbolos s_{215}, s_{714} y s_{298} no pueden representar ningún estado válido.

El alfabeto del sistema es el siguiente conjunto

$$V_1 = \left\{ s_n \quad \forall n \in (Z/4Z)^3 \right\}$$

El conjunto de reglas de producción está formado por todas las posibles transiciones. Así, tras analizar los ejemplos de la sección anterior, se conoce las siguentes tres reglas de producción:

$$s_{444} ::= s_{444}$$

$$s_{113} ::= s_{212}$$

$$s_{303} ::= s_{031}$$

El número total de símbolos válidos es igual al número de variaciones con repetición de cinco elementos ($\{0,1,2,3,4\}$) tomados de tres en tres, que es igual a

$5^3 = 125$.

Determinación de las reglas de producción El conjunto completo de reglas de producción se muestra a continuación

$$P_1 = \left\{ \begin{array}{lllll}
s_{444} ::= s_{444}, & s_{434} ::= s_{344}, & s_{424} ::= s_{433}, & s_{414} ::= s_{332}, & s_{404} ::= s_{042}, \\
s_{344} ::= s_{344}, & s_{334} ::= s_{434}, & s_{324} ::= s_{333}, & s_{314} ::= s_{232}, & s_{304} ::= s_{033}, \\
s_{244} ::= s_{434}, & s_{234} ::= s_{334}, & s_{224} ::= s_{422}, & s_{214} ::= s_{223}, & s_{204} ::= s_{024}, \\
s_{144} ::= s_{334}, & s_{134} ::= s_{423}, & s_{124} ::= s_{413}, & s_{114} ::= s_{214}, & s_{104} ::= s_{014}, \\
s_{044} ::= s_{424}, & s_{034} ::= s_{414}, & s_{024} ::= s_{404}, & s_{014} ::= s_{204}, & s_{004} ::= s_{004}, \\
s_{443} ::= s_{444}, & s_{433} ::= s_{343}, & s_{423} ::= s_{431}, & s_{413} ::= s_{141}, & s_{403} ::= s_{131}, \\
s_{343} ::= s_{344}, & s_{333} ::= s_{432}, & s_{323} ::= s_{331}, & s_{313} ::= s_{041}, & s_{303} ::= s_{031}, \\
s_{243} ::= s_{433}, & s_{233} ::= s_{332}, & s_{223} ::= s_{231}, & s_{213} ::= s_{221}, & s_{203} ::= s_{022}, \\
s_{143} ::= s_{333}, & s_{1333} ::= s_{421}, & s_{123} ::= s_{411}, & s_{113} ::= s_{212}, & s_{103} ::= s_{013}, \\
s_{043} ::= s_{422}, & s_{033} ::= s_{412}, & s_{023} ::= s_{402}, & s_{013} ::= s_{203}, & s_{003} ::= s_{004}, \\
s_{442} ::= s_{443}, & s_{432} ::= s_{342}, & s_{422} ::= s_{241}, & s_{412} ::= s_{231}, & s_{402} ::= s_{221}, \\
s_{342} ::= s_{343}, & s_{332} ::= s_{431}, & s_{322} ::= s_{141}, & s_{312} ::= s_{131}, & s_{302} ::= s_{121}, \\
s_{242} ::= s_{432}, & s_{232} ::= s_{331}, & s_{222} ::= s_{041}, & s_{212} ::= s_{031}, & s_{202} ::= s_{021}, \\
s_{142} ::= s_{332}, & s_{132} ::= s_{231}, & s_{122} ::= s_{221}, & s_{112} ::= s_{211}, & s_{102} ::= s_{012}, \\
s_{042} ::= s_{421}, & s_{032} ::= s_{411}, & s_{022} ::= s_{401}, & s_{012} ::= s_{202}, & s_{002} ::= s_{003}, \\
s_{441} ::= s_{441}, & s_{431} ::= s_{431}, & s_{421} ::= s_{421}, & s_{411} ::= s_{411}, & s_{401} ::= s_{401}, \\
s_{341} ::= s_{341}, & s_{331} ::= s_{331}, & s_{321} ::= s_{321}, & s_{311} ::= s_{311}, & s_{301} ::= s_{301}, \\
s_{241} ::= s_{241}, & s_{231} ::= s_{231}, & s_{221} ::= s_{221}, & s_{211} ::= s_{211}, & s_{201} ::= s_{201}, \\
s_{141} ::= s_{141}, & s_{131} ::= s_{131}, & s_{121} ::= s_{121}, & s_{111} ::= s_{111}, & s_{101} ::= s_{101}, \\
s_{041} ::= s_{041}, & s_{031} ::= s_{031}, & s_{021} ::= s_{021}, & s_{011} ::= s_{011}, & s_{001} ::= s_{001}, \\
s_{440} ::= s_{400}, & s_{430} ::= s_{300}, & s_{420} ::= s_{200}, & s_{410} ::= s_{100}, & s_{400} ::= s_{000}, \\
s_{340} ::= s_{400}, & s_{330} ::= s_{300}, & s_{320} ::= s_{200}, & s_{310} ::= s_{100}, & s_{300} ::= s_{000}, \\
s_{240} ::= s_{400}, & s_{230} ::= s_{300}, & s_{220} ::= s_{200}, & s_{210} ::= s_{100}, & s_{200} ::= s_{000}, \\
s_{140} ::= s_{400}, & s_{130} ::= s_{300}, & s_{120} ::= s_{200}, & s_{110} ::= s_{100}, & s_{100} ::= s_{000}, \\
s_{040} ::= s_{400}, & s_{030} ::= s_{300}, & s_{020} ::= s_{200}, & s_{010} ::= s_{100}, & s_{000} ::= s_{000}
\end{array} \right\}$$

Determinación del axioma Para obtener un axioma adecuado (α_1) se debe deducir los símbolos de las celdas de la rejilla del autómata utilizando la misma función que se ha presentado informalmente en los párrafos anteriores:

$$f\left(a^{\#a}, b^{\#b}, x^{\#x}\right) = s_{\#a\,\#b\,\#x}$$

Expresión del sistema L para el paso de depredación y reproducción Es fácil comprobar que, por construcción, el sistema L

$$(V_1, P_1, \alpha_1)$$

es equivalente al autómata celular descrito para el paso de depredación y reproducción.

El paso de movimiento

En este paso, cada individuo de la celda elige aleatoriamente una dirección de las cuatro posibles. Para distinguir cada individuo por el destino que ha decidido seguir, se utilizará los símbolos del conjunto $\{\uparrow, \rightarrow, \downarrow, \leftarrow\}$. Para cada tipo de individuo se construirá una cadena con los símbolos correspondientes a las direcciones elegidas por cada uno de ellos. Por convenio, serán ordenados de la misma manera en la que aparecen en el conjunto. El contenido de cada celda será representado añadiendo como subíndices las tres cadenas descritas. Por ejemplo, la cadena para una celda en la que hay dos depredadores en estado "a" (uno de ellos ha elegido ir hacia el sur y el otro hacia el este), tres depredadores en estado "b",(ninguno irá hacia el norte) y cuatro presas; es

$$a_{\downarrow\leftarrow}b_{\rightarrow\downarrow\leftarrow}x_{\uparrow\rightarrow\downarrow\leftarrow}.$$

Son necesarias dos fases para simular el movimiento:

- En la primera cada individuo elige aleatoriamente la dirección que seguirá en la siguiente fase.
- En la segunda cada individuo llega a su destino.

La figura 7.b muestra gráficamente otro ejemplo de esta fase. Por claridad, se ha utilizado un símbolo para cada individuo.

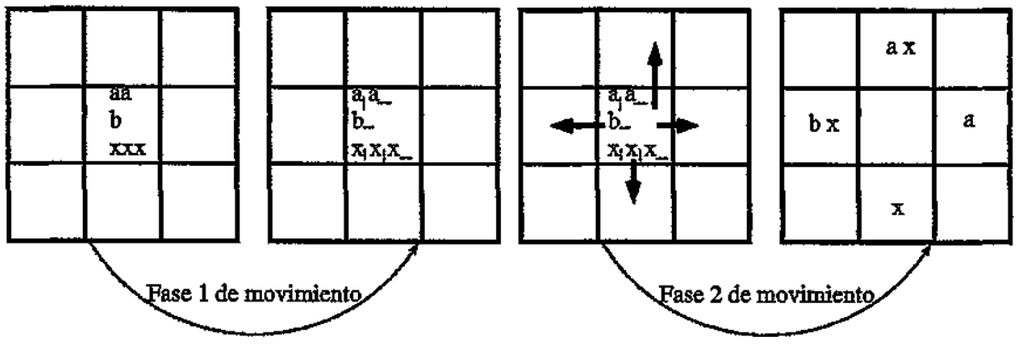


Figura 7.b: Las dos fases de movimiento en el autómata celular.

En el ejemplo anterior, los dos individuos de tipo a podrían elegir aleatoriamente un elemento del siguiente conjunto, que contiene las únicas subcadenas permitidas para los individuos de este tipo:

$$\{a_{\uparrow\rightarrow}, a_{\uparrow\downarrow}, a_{\uparrow\leftarrow}, a_{\rightarrow\downarrow}, a_{\rightarrow\leftarrow}, a_{\downarrow\leftarrow}\}$$

El mismo razonamiento podría aplicarse para los individuos de los otros tipos. Una de las cadenas posibles tras la primera fase para este ejemplo podría ser la siguiente:

$$a_{\downarrow\leftarrow}b_{\rightarrow\downarrow\leftarrow}x_{\uparrow\rightarrow\downarrow\leftarrow}$$

Para continuar el ejemplo en la segunda fase del movimiento se supondrá la situación inicial que muestra la figura 7.c

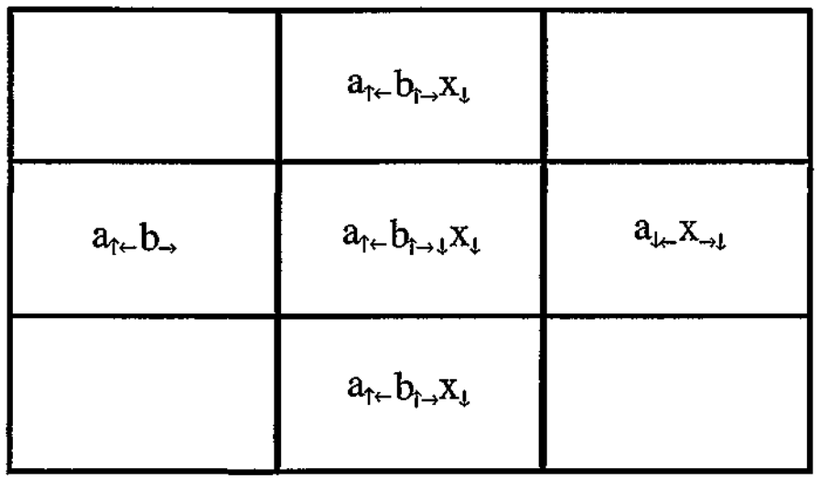


Figura 7.c: Configuración ejemplo para fase 2 de movimiento.

El próximo estado de la celda central sólo depende del valor de las vecinas, en particular, su contenido viene determinado por los individuos cuyas direcciones apuntan a ella. El hecho de que haya cuatro celdas, un único individuo de cada tipo viajando en cada dirección y un máximo de cuatro individuos hace que cualquier situación posible esté permitida. El siguiente estado para la celda de nuestro ejemplo se representa mediante la cadena

$$a_{\uparrow\rightarrow}b_{\rightarrow\downarrow}x_{\downarrow}$$

Sistema L equivalente al autómata del paso de movimiento

Determinación del alfabeto

Es posible construir un sistema L bidimensional cuyas palabras derivadas correspondan a las sucesivas generaciones de este autómata. Es aconsejable utilizar una representación más manejable. Para ello, la palabra formada por los símbolos asociados a las direcciones será sustituida por números de cuatro dígitos binarios. Cada dígito representa la dirección que le corresponde en el orden elegido para el conjunto de direcciones. Un valor 1 en una posición indica que en la celda hay un individuo que ha elegido la dirección asociada a esa posición 1. Un valor 0 indica que ningún individuo del tipo considerado quiere abandonar la celda por esa dirección.

Por ejemplo:

- La cadena 1111 hace referencia a la presencia de cuatro individuos (el tipo depende del símbolo del que esta cadena sea subíndice) y la elección de una dirección distinta por cada uno de ellos.

- La cadena 0101 significa que un individuo de la celda va hacia el este, otro hacia el oeste y ninguno en las direcciones restantes.
- La cadena 1000 significa que sólo hay un individuo que se dirige hacia el norte.

Según esto, al estado de una celda del autómata representado por la cadena

$$a_{\uparrow\downarrow}b_{\uparrow\rightarrow\downarrow\leftarrow}x_{\uparrow\rightarrow\downarrow\leftarrow}$$

le correspondería en el sistema de Lindenmayer equivalente el símbolo

$$s_{1010,1111,0001}$$

De forma que el alfabeto del sistema L se define de la siguiente manera:

$$V_2 = \left\{ s_{n_a,n_b,n_x} \mid \quad n_a, n_b, n_x \in \{0,1\}^4 \right\}$$

Así como en el autómata celular había dos fases, el sistema L tendrá un conjunto de reglas de producción distinto para cada una de ellas. Por lo tanto el sistema de Lindenmayer equivalente será un sistema con dos tablas que se aplicarán siguiendo el mismo comportamiento que en el autómata celular:

Tabla para la fase de elección de dirección En esta fase cada individuo elige aleatoriamente su dirección entre las cuatro posibles con la limitación de que cada dirección puede ser elegida por un único individuo de cada tipo. La tabla de esta fase tiene una enorme cantidad de reglas de producción. Para incrementar la legibilidad del ejemplo, se proporciona un algoritmo que genera la parte derecha de cada regla de producción a partir de la parte izquierda.

Algoritmo Analicemos un ejemplo para aclarar el objetivo del algoritmo.

Sea $s_{1010,1111,0001}$ el símbolo estudiado. El conjunto de reglas de producción con este símbolo como parte izquierda es el siguiente:

$$\left\{ s_{1010,1111,0001} ::= s_{n_1,1111,n_3} \right\}$$

Donde

- n_1 es cualquier permutación de la cadena 1010.
- n_3 es cualquier permutación de la cadena 0001.

Obsérvese que este conjunto tiene cardinal igual a 24.

En general

$$P_2 = \left\{ s_{n_a,n_b,n_x} ::= s_{\widehat{n_a},\widehat{n_b},\widehat{n_x}} \quad \begin{array}{c} \forall s_{n_a,n_b,n_x} \in V_2, \\ \forall \widehat{n_a} \text{ permtcn } de\ n_a, \\ \forall \widehat{n_b} \text{ permtcn } de\ n_b, \\ \forall \widehat{n_x} \text{ permtcn } de\ n_x \end{array} \right\}$$

P_2 es obviamente no determinista.

Tabla para la fase de ejecución del movimiento Una vez que cada individuo sabe la dirección en la que se moverá, se aplica la segunda tabla. Esta tabla formaliza el movimiento que cada individuo realiza para seguir la dirección que ha seleccionado en la fase anterior. La clave para definir la tabla consiste en cambiar el punto de vista. Las transformaciones formalizadas informarán del cambio en el contenido de una celda no en función de lo que sale de ella sino de lo que le llega desde sus vecinos. La figura 7.d muestra gráficamente este cambio de punto de vista. Compárese esta figura con la figura 7.b para apreciar este aspecto con más claridad. Las interrogaciones del dibujo de la derecha representan celdas cuyo contenido no se conoce al depender del estado del resto de la rejilla no representado en la figura.

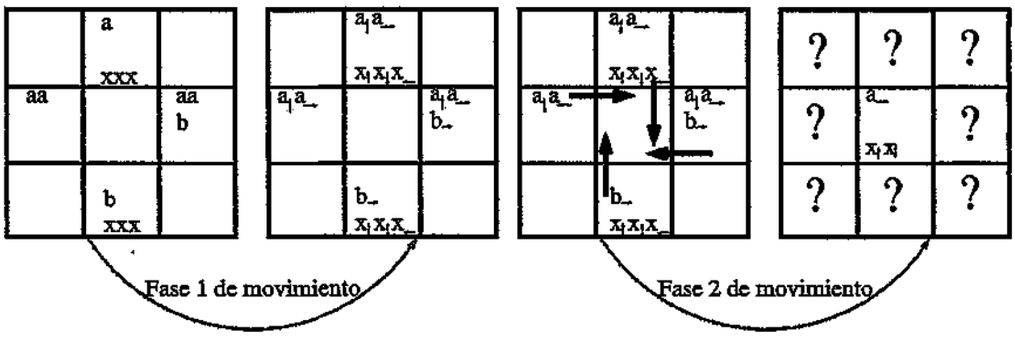


Figura 7.d: Cambio de punto de vista para la determinación de la segunda tabla.

Dado que hay 4 direcciones permitidas, será necesario considerar los 4 vecinos más próximos que rodeen la celda estudiada. Por lo tanto esta tabla tiene interacciones y una vecindad IL de von Neumann.

Algoritmo para la construcción de la tabla para la fase de ejecución del movimiento A pesar del gran número de reglas, esta tabla es determinista. Para obtener el símbolo correspondiente al nuevo estado de una celda, es suficiente formar cada uno de los tres números de su subíndice de la siguiente manera:

- El primer dígito del resultado se toma del vecino inferior.
- El segundo dígito del vecino izquierdo.
- El tercer dígito del vecino superior.
- El cuarto dígito del vecino derecho.

La gráfica 7.e muestra gráficamente este algoritmo.

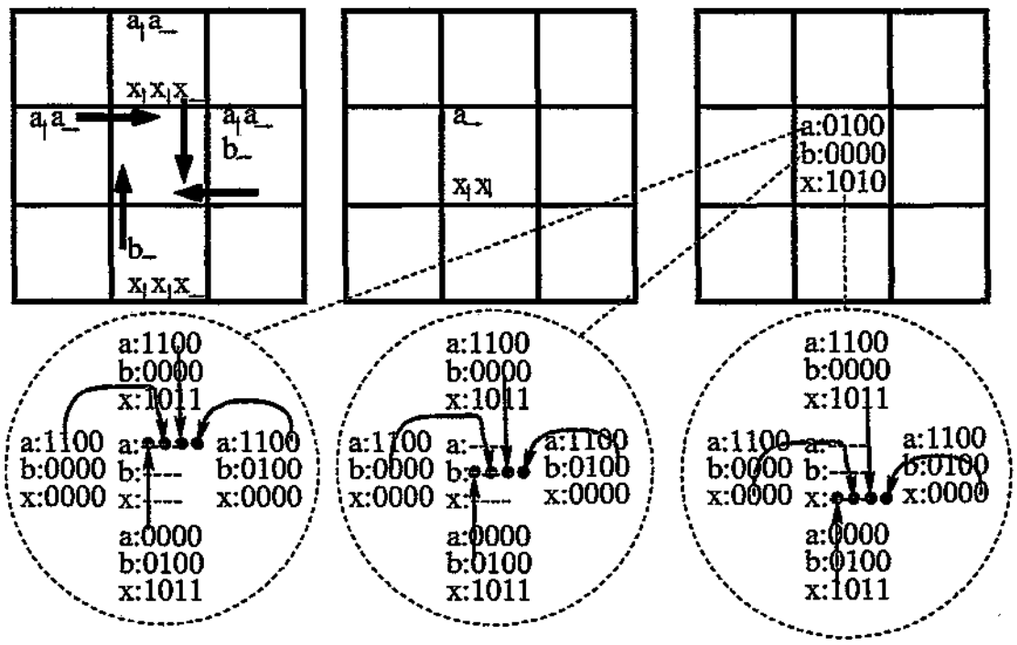


Figura 7.e: Ejemplo de determinación de una regla para la tabla de la fase de ejecución del movimiento.

En general el conjunto de reglas de producción puede definirse de la siguiente manera:

$$P_3 = \left\{ \begin{array}{c} n_{n_a^n,n_b^n,n_x^n} e_{n_a^e,n_b^e,n_x^e} s_{n_a^s,n_b^s,n_x^s} o_{n_a^o,n_b^o,n_x^o} s_{n_a,n_b,n_x} ::= s_{\widehat{n_a},\widehat{n_b},\widehat{n_x}} \\ \forall n_a^n, n_b^n, n_x^n, n_a^e, n_b^e, n_x^e, n_a^s, n_b^s, n_x^s, n_a^o, n_b^o, n_x^o, n_a, n_b, n_x \in \{0,1\}^4 \mid \\ \widehat{n_a} = n_a^s n_a^e n_a^n n_a^o \\ \widehat{n_b} = n_b^s n_b^e n_b^n n_b^o \\ \widehat{n_x} = n_x^s n_x^e n_x^n n_x^o \end{array} \right\}$$

Donde

- $n_{n_a^n,n_b^n,n_x^n}$, es el símbolo del vecino superior (*n*orte).
- $e_{n_a^e,n_b^e,n_x^e}$, es el símbolo del vecino de la derecha (*e*ste).
- $s_{n_a^s,n_b^s,n_x^s}$, es el símbolo del vecino inferior (*s*ur).
- $o_{n_a^o,n_b^o,n_x^o}$, es el símbolo del vecino izquierdo (*o*este).
- s_{n_a,n_b,n_x}, es el símbolo transformado por la regla.

Determinación del axioma El axioma (α_2) para este sistema es la rejilla bidimensional infinita sobre V_2 obtenida al expresar el estado de cada nodo de la rejilla del autómata con los convenios explicados en los párrafos precedentes.

Expresión del sistema L para el paso de movimiento Es fácil comprobar que el sistema L

$$(V_2, \{P_2, P_3\}, g, \alpha_2, V_{LN})$$

es equivalente al autómata celular de movimiento.

Donde

- V_2, P_1,P_2,α_2 han sido definidos en esta misma sección.
- g es el símbolo de marca del sistema IL.
- V_{LN} es la vecindad bidimensional IL de von Neumann.

9.4.2 El sistema L equivalente a la combinación de los dos autómatas celulares del ecosistema

Para poder formar un único sistema L a partir de los dos anteriores es necesario introducir alguna información redundante en los símbolos del alfabeto para unificar las notaciones de V_1 y V_2.

Tras construir el nuevo alfabeto, las tablas de producción deberán ser redefinidas para que manejen la nueva notación.

Al analizar los dos tipos de subíndices distintos utilizados en V_1 y V_2, se comprueba que los utilizados en V_1 pueden ser obtenidos como el número de dígitos 1 que hay en los usados en V_2. Una opción que permite redefinir fácilmente los conjuntos de reglas de producción es utilizar simultáneamente los dos tipos de subíndice de forma que cada tabla utilice los que necesite. Uno de los dos conjuntos de subíndices es redundatne. De esa manera el alfabeto final se puede definir de la siguiente forma:

$$V_3 = \left\{ s_{n_a,n_b,n_x,u_a,u_b,u_x} \mid \quad n_a, n_b, n_x \in \{0,1\}^4, u_i = apariciones(1, n_i) \quad \forall i \in \{a,b,x\} \right\}$$

Donde *apariciones*$(letra, cadena)$ es la función definida en el capítulo 6 que devuelve el número de veces que el símbolo que es su primer argumento aparece en la cadena que es su segundo argumento.

Los conjuntos de reglas de producción se redefinen de manera que los elementos de P_1 utilicen como subíndices $u_a u_b u_x$ y los de P_2 y P_3 $n_a n_b n_x$. Llamaremos respectivamente $\widehat{P_1}$, $\widehat{P_2}$ y $\widehat{P_3}$ a los conjuntos de reglas de producción modificados de esta manera.

El axioma también tiene que ser modificado conforme a la redefinición del alfabeto. El proceso de cálculo de los dos juegos de índices es el mismo que se explicó en las secciones anteriores para calcular α_1 y α_2.

Es fácil comprobar que el sistema $\langle 4,0 \rangle$ IL

$$\left(V_3, \left\{\widehat{P_1}, \widehat{P_2}, \widehat{P_3}\right\}, g, \alpha_3, V_{LN}\right)$$

Es equivalente al autómata estudiado.

Sobre la eficiencia

Puede compararse la complejidad de los algoritmos para simular el paso de reproducción y depredación del ecosistema sugeridos por ambos modelos: el autómata celular y el sistema de Lindenmayer equivalente.

El autómata celular necesita un bucle para recorrer todos los autómatas finitos de su rejilla. Para determinar el siguiente estado necesita un bucle que recorra todos sus vecinos. Se puede deducir que la complejidad de la simulación mediante el autómata celular es $o(n) \simeq k_1 n$ donde n es el número de autómatas finitos de la rejilla.

El sistema de Lindenmayer necesita un bucle para recorrer todos los símbolos de la palabra pero, determina el símbolo derivado accediendo directamente a una tabla. Se puede deducir que la complejidad de la simulación mediante el sistema L es $o(n) \simeq k_2 n$ donde n es el tamaño de las palabras.

Las pruebas realizadas han mostrado una reducción sensible en el tiempo de ejecución de la simulación mediante el sistema de Lindenmayer lo que sugiere que se puede conseguir implementaciones en las que se mejore la constante.

9.5 Un autómatas celular tridimensional que genera y propaga un "pulso"

Se utilizará como rejilla tridimensional un prisma potencialmente infinito hacia la derecha y compuesto por pequeños cubos de d unidades de lado de manera que cualquier sección perpendicular del prisma corta cuatro cubos.

La figura 7.f muestra un esquema de la rejilla.

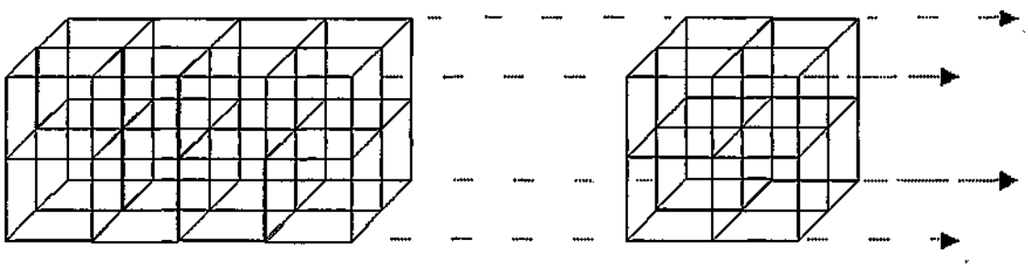


Figura 7.f : Rejilla.

La figura 7.g muestra la vecindad tridimensional de von Neumann que es la utilizada por el ejemplo.

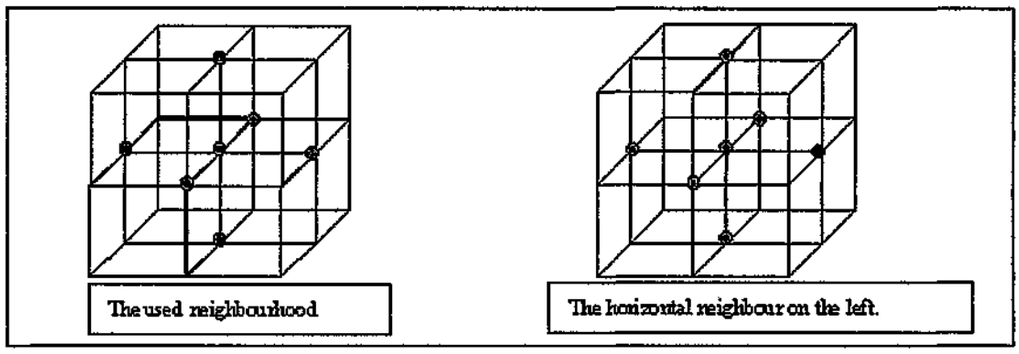


Figura 7. g: Vecindad.

Es decir, la vecindad de una celda está formada por ella misma y sus 6 vecinas más próximas, las 6 celdas que la rodean a una distancia de d unidades.

El conjunto de estados del autómata es $\{0, 1\}$. La configuración inicial de la rejilla se muestra en la figura 7.h.

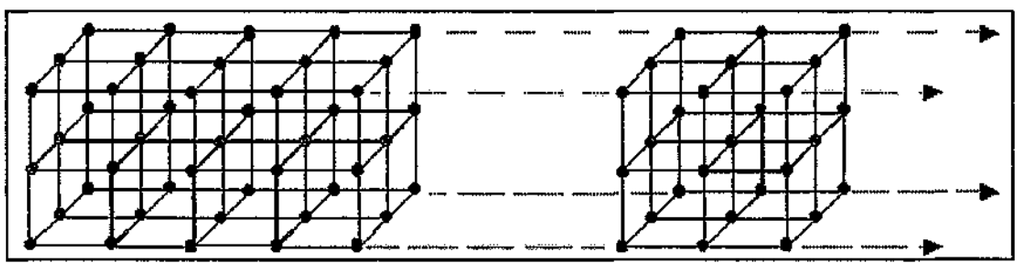


Figura 7. h: Configuración inicial.

En esta figura, los puntos oscuros representan autómatas con estado 1 y los puntos claros autómatas con estado 0. Cualquier valor de estado necesario y no representado explícitamente en esta figura se supone por convenio que es 0. Estos valores se utilizarán, por ejemplo para calcular el siguiente estado de los autómatas de los bordes del prisma que dependerá de unos vecinos que no se han representado en la gráfica y que, por tanto, tomarán inicialmente el valor 0.

Para calcular el siguiente estado, cada autómata de la rejilla sigue las siguientes reglas:

- No se tiene en cuenta en ningún caso el vecino horizontal de la derecha (el que está resaltado en la parte derecha de la figura 7.g donde se muestra la vecindad).
- Si los cuatro vecinos del plano vertical (excluída la posición ocupada por el autómata considerado) tienen estado 0, el siguiente estado del autómata estudiado cambia su valor. En este caso, el valor de su vecino de la izquierda no es tenido en cuenta. La tabla siguiente y la figura 7.i muestran esta regla de comportamiento.

Vecinos						*Estado i del actual*	*Estado i + 1 del actual*
Superior	*Inferior*	*Frente*	*Espalda*	*Derecha*	*Izquierda*		
0	0	0	0	x	y	0	1
0	0	0	0	x	y	1	0

Donde x e y representan cualquier valor del conjunto $\{0,1\}$.

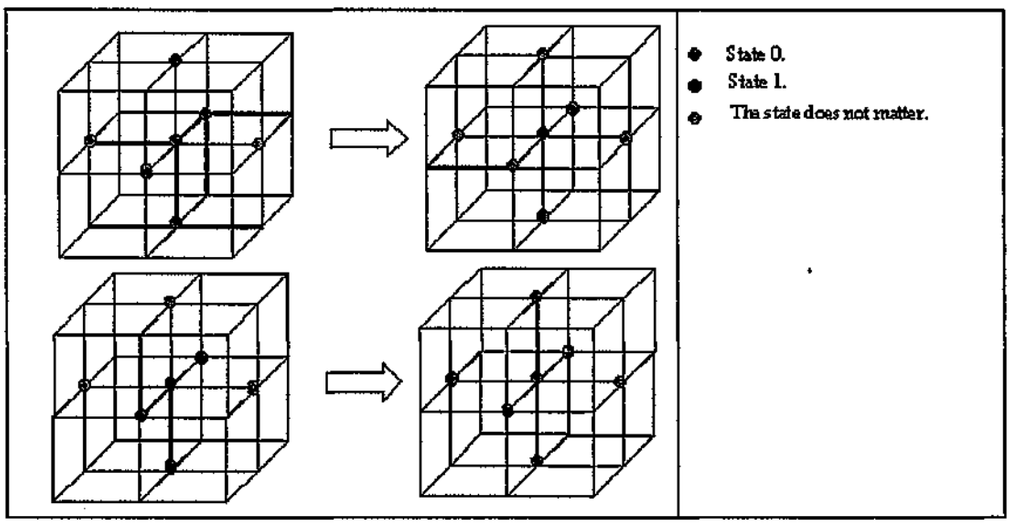


Figura 7. i: Función de transición del autómata: anillo vertical de vecindario a 0.

- Si los cuatro vecinos del plano vertical (excluída la posición ocupada por el autómata considerado) tienen estado 1, el siguiente estado del autómata actual depende del estado de su vecino izquierdo y de su propio estado. El autómata cambiará de estado cuando ambos valores difieran y en otro caso quedará inalterado. La tabla siguiente y la figura 7.j muestran esta regla de comportamiento.

Vecinos						*Estado i del actual*	*Estado i + 1 del actual*
Superior	*Inferior*	*Frente*	*Espalda*	*Derecha*	*Izquierda*		
1	1	1	1	x	0	0	0
1	1	1	1	x	0	1	0
1	1	1	1	x	1	0	1
1	1	1	1	x	1	1	1

Donde x representa cualquier valor del conjunto $\{0,1\}$. En cualquier otro caso no contenido en la tabla el valor del estado del autómata permanece inalterado.

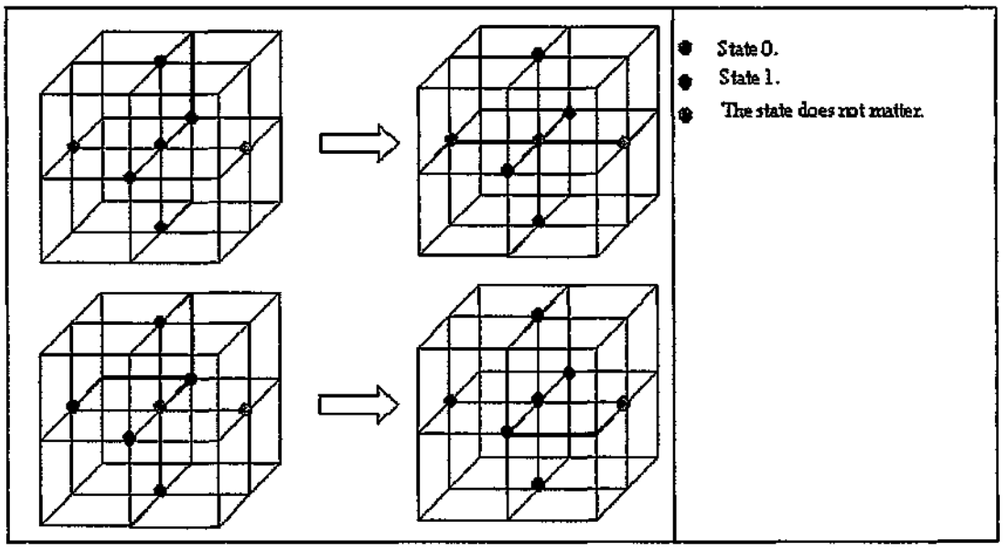


Figura 7. j: Función de transición del autómata: anillo vertical de vecindario a 1.

La figura 7.k muestra el comportamiento del autómata mediante los primeros pasos tras la configuración inicial.

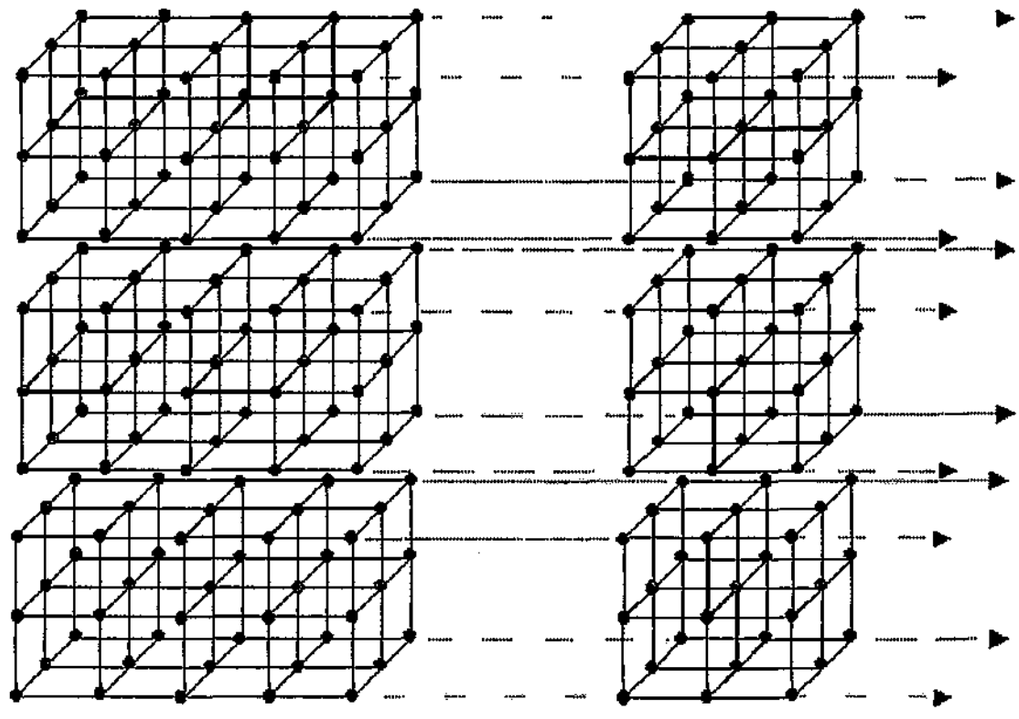


Figura 7. k: Conducta del autómata.

Obsérvese cómo la cara izquierda del prisma genera pulsos en los pasos impares, que son propagados a través del eje central del prisma.

Los estados iniciales para los autómatas de los vértices de la cara del extremo izquierdo son especialmente relevantes. Si su valor es 0, varios autómatas de esta cara cambiarían su valor generando pulsos espúreos que distorsionarían la propagación del pulso central. El objetivo de este autómata celular es generar y propagar pulsos a lo largo de su eje central. El estado de todos los autómatas de los bordes del prisma permanece inalterado porque sus vecinos del plano vertical nunca tienen el mismo estado.

Se puede construir un sistema IL tridimensional cuyas palabras derivadas correspondan con las sucesivas generaciones del autómata. La vecindad utilizada será la IL de von Neumann tridimensional. Esta vecindad coincide con la dibujada para el autómata celular pero excluye la posición de la rejilla estudiada. El alfabeto de este sistema es el conjunto $V_4 = \{0, 1\}$.

Para presentar de manera legible el contexto se utilizará el siguiente orden que identifica cada vecino en las partes izquierdas de cada regla de producción:

Orden	*Celda estudiada*
0	*Vecino superior*
1	*Vecino inferior*
2	*Vecino frontal*
3	*Vecino trasero*
4	*Vecino derecho*
5	*Vecino izquierdo*
6	*Celda estudiada*

Las reglas de producción tendrán como parte izquierda un vector binario cuyos dígitos son los estados de los siete autómatas. Se puede copiar las reglas directamente de las tablas usadas para describir la función de transición del autómata celular. El conjunto de reglas puede describirse de la siguiente manera:

$$P_4 = \left\{ \begin{array}{l} 0000xy0 ::= 1, \\ 0000xy1 ::= 0, \\ 1111x00 ::= 0, \\ 1111x01 ::= 0, \\ 1111x10 ::= 1, \\ 1111x11 ::= 1 \end{array} \quad \forall x, y \in V_4 \right\} \cup$$
$$\{xyzuvws ::= s \quad \forall xyzu \notin \{0000, 1111\}, \quad \forall v, w, s \in V_4\}$$

El axioma α_4 es la rejilla tridimensional binaria obtenida a partir de la figura 7.h de manera que el valor de cada posición se obtiene aplicando la siguiente regla:

$$\left\{ \begin{array}{lll} 1 & si & \textit{el color del punto es oscuro} \\ 0 & otro & caso \end{array} \right.$$

Es fácil comprobar que el sistema de Lindenmayer $\{V_4, P_4, \alpha_4\}$ es equivalente al autómata tridimensional estudiado.

Capítulo 10

Representación de autómatas celulares probabilistas mediante sistemas L

10.1 Descripción intuitiva: sistemas IL n-dimensionales

Habitualmente las palabras derivadas por los sistemas L se disponen en forma de cadenas. Aunque se podría seguir utilizando esa representación, en ciertos procesos es más cómodo disponer los símbolos de otra manera. Muchos procesos pueden describirse geométricamente en base a matrices, retículos, etc... En este trabajo se dispondrá las palabras de los sistemas IL en forma de rejilla n-dimensional.

10.2 Definición

Un sistema IL n-dimensional es un sistema IL cuyas palabras son rejillas de caracteres en lugar de cadenas lineales. Las rejillas fueron definidas en el capítulo 4.

Para simplificar la notación se utilizarán los siguientes convenios:

- En la parte izquierda de las reglas de producción, el contexto siempre se escribirá antes de que el símbolo transformado por las reglas.
- Se utilizará una vecindad n-dimensional para localizar en la rejilla cada uno de los símbolos del contexto.

Formalmente,

Un *sistema* $(k,0)IL$ *n-dimensional* es una quíntupla

$$(\sum, P, g, w, c)$$

Donde

- $\sum, P, g$ han sido definidos en la sección 3.2.5.
- w (el axioma) es una rejilla n-dimensional sobre $\sum$.
- c es una vecindad n-dimensional de k vecinos, es decir $c = (k, N)$ y N (el vector de desplazamientos) se determina en cada caso.

10.3 Notación

Cuando sea más cómodo referirse a las componentes de un sistema $(k,0)$IL n-dimensional G en función del propio sistema se utilizará la siguiente notación (sólo se añade componentes no definidas previamente).

- *vecindad* (G)
- *LongitudContexto* $(G) = k$

10.4 Definición

La vecindad bidimensional IL de von Neumann, que se escribe V_{LN}, es la vecindad bidimensional que elige los cuatro vecinos más próximos según la distancia euclídea.

Formalmente,

- $V_{LN} = (4, ((0,1), (1,0), (0,-1), (-1,0)))$

La figura 7.0.f muestra la disposición de esta vecindad en la rejilla :

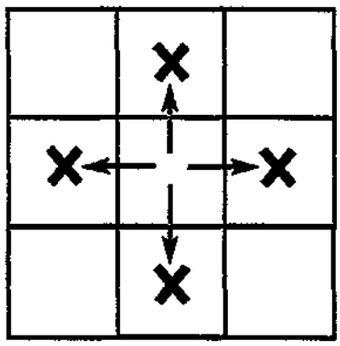


Figura 7.0.f Vecindad bidimensional IL de von Neumann de la posición central.

Obsérvese que la vecindad IL de von Neumann es la de von Neumann excluyendo la posición estudiada.

10.5 Definición

La vecindad bidimensional IL de Moore, que se escribe V_{LM}, es la vecindad bidimensional que elige los ocho vecinos más próximos a ella según la distancia euclídea.

Formalmente,

- $V_{LM} = (8, ((-1,1),(0,1),(1,1),(1,0),(1,-1),(0,-1),(-1,-1),(-1,0)))$

La gráfica 7.0.g muestra la disposición de esta vecindad en la rejilla:

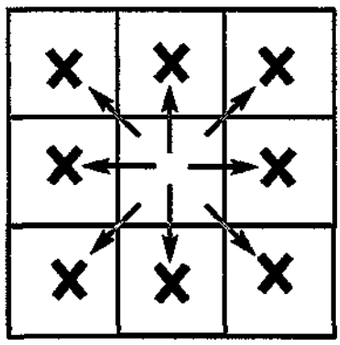


Figura 7.0.g Vecindad IL de Moore de la posición central.

Obsérvese que la vecindad IL de Moore es la de Moore excluyendo la posición considerada.

10.6 Descripción informal: sistemas L probabilistas

Los sistemas L descritos hasta ahora se han podido clasificar de la siguiente manera atendiendo a la forma de sus reglas de producción:

- Deterministas, cuando a cada parte izquierda le corresponde exclusivamente una parte derecha.
- No determinista, hay al menos una parte izquierda a la que le corresponde un conjunto de posibles partes derechas.

Introducimos los sistemas L probabilistas, que asignan explícitamente a cada parte derecha una probabilidad de ser elegida en cada derivación posible.

Desde este punto de vista los sistemas L deterministas pueden ser vistos como casos particulares de sistemas L probabilistas, pues asignan implícitamente a sus únicas partes derechas probabilidad igual a 1.

10.7 Notación

Para toda regla de producción para un sistema L ($R = x ::= y$) hay un único símbolo en su parte izquierda que será sustituido por la parte derecha de la regla. Escribiremos *sustituible* (x) para representar ese símbolo.

10.8 Definicion, conjunto de reglas de producción probabilista

Un *conjunto de reglas de producción probabilista* P^Pes un conjunto de pares en los que una componente es una regla de producción y la otra es su probabilidad. Es obligatorio que la suma de las probabilidades asignadas a todas las reglas aplicables al mismo símbolo sea igual a 1.

Y los elementos de P^P se escribirán

$$(R, p(R))$$

Es decir

- R es la regla.
- $p(R)$ es su probabilidad

10.9 Definicion, sistema L probabilista

Un *sistema L probabilista* es un sistema L que en lugar de tener un conjunto de reglas de producción tiene un conjunto de reglas de producción probabilista.

10.10 Descripción informal: árbol de derivación de una cadena en un esquema L no determinista

En los sistemas de Lindenmayer deterministas, las derivaciones son lineales en el sentido de que la derivación de una cadena sólo puede ser una única cadena.

En los sistemas de Lindenmayer no deterministas o probabilistas hay varias cadenas que son posibles derivaciones de cada palabra.

En estos casos, la representación gráfica de una derivación adopta la forma de un árbol.

10.11 Definición, árboles de derivación

Sea S un esquema L no determinista.

Sea s una cadena no vacía de su alfabeto.

Sea n un número natural.

El *árbol de profundidad n para una cadena s en el esquema S*, que se representará $T_n(s,S)$ o simplemente $T_n(s)$ si no hay ambigüedad respecto al esquema considerado, es el conjunto de todas las posibles derivaciones de profundidad n que pueden ser construidas mediante el esquema S a partir de la cadena s.

Formalmente

$$S = (\Sigma, P)$$

$$s \in \Sigma^+$$

$$n \in \aleph$$

$$T_n(s,S) = \{D | \quad prof(D) = n \quad \wedge \quad \pi_0(tr(D)) = s\}$$

10.12 Representación gráfica de $T_n(s,S)$

En el presente trabajo se utilizará la representación arbórea del conjunto $T_n(s,S)$ que se consigue mediante la aplicación del algoritmo que llamaremos.

$Grafo(n,s,S)$

Donde

- n es la profundidad del árbol.
- s es la cadena de partida.
- $S = (\Sigma, P^P)$ es el esquema probabilista de Lindenmayer utilizado.

En el algoritmo se construirá una representación gráfica arbórea en base a nodos y arcos entre ellos. La representación arbórea, por tanto, será un par de conjuntos, uno de ellos de nodos y el otro de arcos o parejas de nodos etiquetados. A esa representación se la llamará *grafo*.

Para construir esa representación se utilzará la siguiente información.

- Derivaciones con la notación descrita en la sección 3.2.2, es decir, una derivación D en S es una terna

$$(O \in \aleph \times \aleph, \quad v : O \to \Sigma, \quad p : O \to P^P)$$

- Los arcos estarán entiquetados con las componentes v y p de la derivación de un paso que transforme una palabra en la siguiente.
- El contenido de cada nodo de la representación arbórea será un par cuya primera componente es una palabra derivable por S desde s y la segunda la longitud o profundidad de la derivación.
- Una lista (tupla sin especificar el tamaño) de nodos pendientes de ser tratados y que se llamará *NodosPorTratar*. Para manipular esta lista se utilizará, además de la notación usual para el acceso a las componentes mediente índice, las siguientes funciones:

- *primero* (*lista*) que devuelve el primer elemento de la lista.
- *resto* (*lista*) que devuelve la lista obtenida a partir de *lista* excluyendo su primer elemento.
- *concatenar* ($lista_1, lista_2$) que genera la lista obtenida con los elementos de $lista_1$ en el mismo orden en el que aparecen y después los elementos de $lista_2$ en el mismo orden en el que aparecen.

- El nodo que se esté tratando en cada momento se llamará *NodoActual*.
- La profundidad contenida en ese nodo se llamará $profundidad$
- La cadena contenida en el *NodoActual* se llamará *cadena*.
- El símbolo del punto ". " se utilizará para representar la concatenación de cadenas.

Tras las observaciones anteriores puede describirse formalmente la representación gráfica que este algoritmo construye.

$$grafo = (nodos, arcos)\,,\, arcos \subseteq nodos \times nodos \times (v : O \to \Sigma \times p : O \to P^P)$$

A continuación se describe el algoritmo.

$Grafo(n, s, S)$

- Crear un nodo cuyo contenido sea $(s, 0)$.
- Dar como valor inicial a la lista de nodos *NodosPorTratar* el nodo $(s, 0)$.
- Dar como valor inicial a la representación que se está construyendo un par formado por un conjunto vacío de nodos y un conjunto vacío de arcos:
  - $grafo \leftarrow (nodos = \Phi, arcos = \Phi)$
- MIENTRAS *NodosPorTratar*$\neq \Phi$ HACER
  - *NodoActual* $\leftarrow$ *primero* (*NodosPorTratar*)
  - *nodos* $\leftarrow$ *nodos* $\cup$ *NodoActual* (se añade el nodo a la representación que se está construyenco)
  - *NodosPorTratar* $\leftarrow$ *resto* (*NodosPorTratar*)
  - $profundidad \leftarrow NodoActual[1]$
  - SI $profundidad < n$ (la profundidad del nodo tratado no es la máxima)
    - ENTONCES
      - *cadena* $\leftarrow$ *NodoActual*[0]
      - $m \leftarrow |NodoActual[0]|$
      - Se construye los m conjuntos de reglas de producción aplicables a cada símbolo de la cadena del nodo actual *cadena* que se pueden describir de la siguiente manera

$$\left\{P_i^P = \left\{r = x_r ::= y_r | r \in P^P \wedge sustituible\,(x_r) = cadena\,[i]\right\}\right\}_{i=0}^{m-1}.$$

- Cada elemento de $P_0^P \times ... \times P_m^P$ es una imagen válida para la función p de una derivación parcial de sólo un paso que obtiene de la cadena del nodo (*cadena*) una posible derivación directa. Es necesario crear un nodo para cada una de esas derivaciones directas incrementando en uno la profundidad del nodo padre y creando un arco que los una que tenga como etiquetas las componentes v y p de la derivación que se está construyendo.

$$NodosPorTratar \leftarrow concatenar\,(NodosPorTratar, NuevosNodos)\ ,,$$
$$NuevosNodos \text{ lista de nodos} \mid \exists i \in \aleph \quad \forall$$
$$(r_0 = x_0 ::= y_0, ..., r_{m-1} = x_{m-1} ::= y_{m-1}) \in P_0^P \times ... \times P_{m-1}^P\ NuevosNodos\,[i]$$
$$= (y_0.y_1.\cdots.y_{m-1},\quad profundidad + 1)$$

- Se añade al conjunto de arcos de la representación uno por cada elemento de *NuevosNodos*, cada arco tendrá como primer nodo el padre de éstos (*NodoActual*) como segundo nodo el elemento de *NuevosNodos* y como etiqueta el subconjunto de la componente v y p de la derivación de la rama que corresponda a la derivación directa representada por el arco. Formalmente

$$arcos \leftarrow arcos \cup \left\{\left(\begin{array}{l} NodoActual, \\ (y_0.y_1.\ \ .y_{m-1},\quad profundidad+1), \\ \left(\begin{array}{l} \{v\,(profundidad, l) = cadena_l\}_{l=0}^{m-1} \\ \{p\,(profundidad, l) = x_l ::= y_l\}_{l=0}^{m-1} \end{array}\right) \end{array}\right)\right\}$$

- El significado de todos los símbolos de la expresión anterior puede deducirse de los puntos anteriores porque hay un arco para cada elemento de *NuevosNodos* y todos esos elementos se han descrito con detalle previamente.

- FIN MIENTRAS
- Devolver $grafo$.

10.13 Descripción informal: cálculo de la probabilidad de que un esquema L probabilista obtenga mediante una derivación de n pasos una cadena a partir de otra

Los árboles de derivación se pueden utilizar para calcular la probabilidad de que se obtenga una cadena x a partir de otra y mediante una derivación de n pasos por un esquema L probabilista S.

Para ello son necesarios los siguientes pasos.

- Construir el árbol de derivación $T_n\,(y, S)$ (o su representación gráfica $Grafo\,(n, y, S)$).
- Sumar la probabilidad de obtener la cadena x para cada elemento del conjunto de ramas cuyas hojas contengan la cadena x.
- En cada una de las ramas, esa probabilidad se calcula multiplicando la probabilidad de cada uno de los arcos.

- Cada arco consiste en la aplicación simultánea de las reglas de producción recogidas en la componente p de su etiqueta. Por lo tanto la probabilidad de cada arco se obtiene multiplicando las probabilidades de esas reglas, que se obtienen directamente del conjunto de reglas de producción probabilista del esquema S (P^P).

Obsérvese que en los dos últimos puntos se menciona la necesidad de realizar dos productos de probabilidades:

- Uno de ellos recorriendo toda la rama de cada derivación y tomando el valor obtenido en cada arco.
- El otro tomando las probabilidades de las reglas aplicadas dentro del mismo arco.

Estos dos productos se reducen a uno si hacemos el recorrido siguiendo la componente O de la derivación.

Por otro lado, las probabilidades alcanzadas para cada cadena tienen que cumplir que la suma de todas las de la misma profundidad en el árbol sea igual a 1.

10.14 Ejemplo

Sea el sistem de $\langle 1,1\rangle$ IL unidimensional siguiente:

$$SL^1 = \left\{ \begin{array}{c} \Sigma^1 = \{0,1\}, \\ P^1 = \left\{ \begin{array}{ll} (xsg ::= x, 1) & \forall x, s \in \Sigma^1, \\ (gsx ::= x, 1) & \forall x, s \in \Sigma^1, \\ (x0y ::= x, 0.3) & \forall x, y \in \Sigma^1, \\ (x0y ::= y, 0.7) & \forall x, y \in \Sigma^1, \\ (x1y ::= y, 0.3) & \forall x, y \in \Sigma^1, \\ (x1y ::= x, 0.7) & \forall x, y \in \Sigma^1 \end{array} \right\}, \\ g, \\ 0011 \end{array} \right\}$$

Se quiere calcular la probabilidad de que los símbolos 0 desaparezcan.

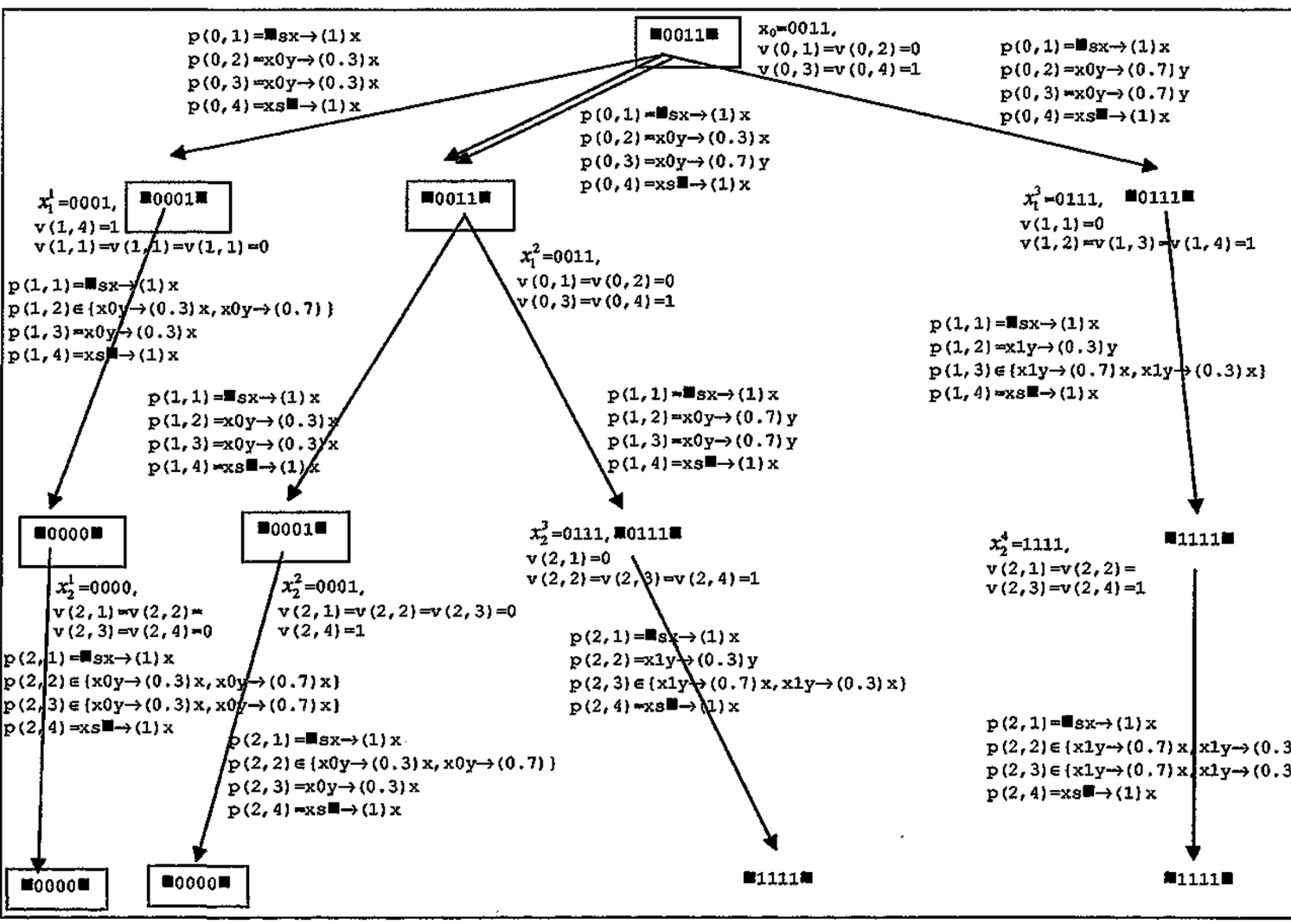
Se quiere también calcular la probabilidad de que los símbolos 1 desaparezcan.

Se quiere comparar ambas probabilidades.

La figura muestra las ramas del árbol de derivación para esa cadena necesarias para analizar esos casos. Para facilitar la lectura se ha utilizado la misma notación para todas las derivaciones del árbol.

10.15 Definición

La *probabilidad de que a partir de una cadena y un esquema probabilista de Lindenmayer S se genere una cadena x en n pasos* se identificará mediante la expresión $p_{y,n,s}(x)$ y es el valor que se calcula mediante la siguiente expresión.


■0011■
x0=0011,
v(0,1)=v(0,2)=0
v(0,3)=v(0,4)=1
p(0,1)=■sx→(1)x
p(0,2)=x0y→(0.3)x
p(0,3)=x0y→(0.3)x
p(0,4)=xs■→(1)x
p(0,1)=■sx→(1)x
p(0,2)=x0y→(0.3)x
p(0,3)=x0y→(0.7)y
p(0,4)=xs■→(1)x
p(0,1)=■sx→(1)x
p(0,2)=x0y→(0.7)y
p(0,3)=x0y→(0.7)y
p(0,4)=xs■→(1)x
■0001■
x11=0001,
v(1,4)=1
v(1,1)=v(1,1)=v(1,1)=0
■0011■
x12=0011,
v(0,1)=v(0,2)=0
v(0,3)=v(0,4)=1
■0111■
x13=0111,
v(1,1)=0
v(1,2)=v(1,3)=v(1,4)=1
p(1,1)=■sx→(1)x
p(1,2)∈{x0y→(0.3)x,x0y→(0.7)}
p(1,3)=x0y→(0.3)x
p(1,4)=xs■→(1)x
p(1,1)=■sx→(1)x
p(1,2)=x0y→(0.3)x
p(1,3)=x0y→(0.3)x
p(1,4)=xs■→(1)x
p(1,1)=■sx→(1)x
p(1,2)=x0y→(0.7)y
p(1,3)=x0y→(0.7)y
p(1,4)=xs■→(1)x
p(1,1)=■sx→(1)x
p(1,2)=x1y→(0.3)y
p(1,3)∈{x1y→(0.7)x,x1y→(0.3)x}
p(1,4)=xs■→(1)x
■0000■
■0001■
x21=0000,
v(2,1)=v(2,2)=
v(2,3)=v(2,4)=0
x22=0001,
v(2,1)=v(2,2)=v(2,3)=0
v(2,4)=1
x23=0111,■0111■
v(2,1)=0
v(2,2)=v(2,3)=v(2,4)=1
■1111■
x24=1111,
v(2,1)=v(2,2)=
v(2,3)=v(2,4)=1
p(2,1)=■sx→(1)x
p(2,2)∈{x0y→(0.3)x,x0y→(0.7)x}
p(2,3)∈{x0y→(0.3)x,x0y→(0.7)x}
p(2,4)=xs■→(1)x
p(2,1)=■sx→(1)x
p(2,2)∈{x0y→(0.3)x,x0y→(0.7)}
p(2,3)=x0y→(0.3)x
p(2,4)=xs■→(1)x
p(2,1)=■sx→(1)x
p(2,2)=x1y→(0.3)y
p(2,3)∈{x1y→(0.7)x,x1y→(0.3)x}
p(2,4)=xs■→(1)x
p(2,1)=■sx→(1)x
p(2,2)∈{x1y→(0.7)x,x1y→(0.3
p(2,3)∈{x1y→(0.7)x,x1y→(0.3
p(2,4)=xs■→(1)x
■0000■
■0000■
■1111■
■1111■

$$p_{y,n,s}(x) = \sum_{\rho(D_i)=x \wedge T_n(y,S)} \left(\prod_{(q,r)\in O_i} p(p_i(q,r)) \right)$$

10.16 Definición

La *probabilidad de que un sistema probabilista de Lindenmayer G genere una cadena x en n pasos* se identificará mediante la expresión $p_{n,G}(x)$ y es la probabilidad de que la genere su esquema a partir de su axioma.

Formalmente

$$p_{n,G}(x) = p_{axioma(G),n,esquema(G)}(x)$$

En la figura 7.0.h se muestra un ejemplo del cálculo de las probabilidades a partir del árbol de derivación.

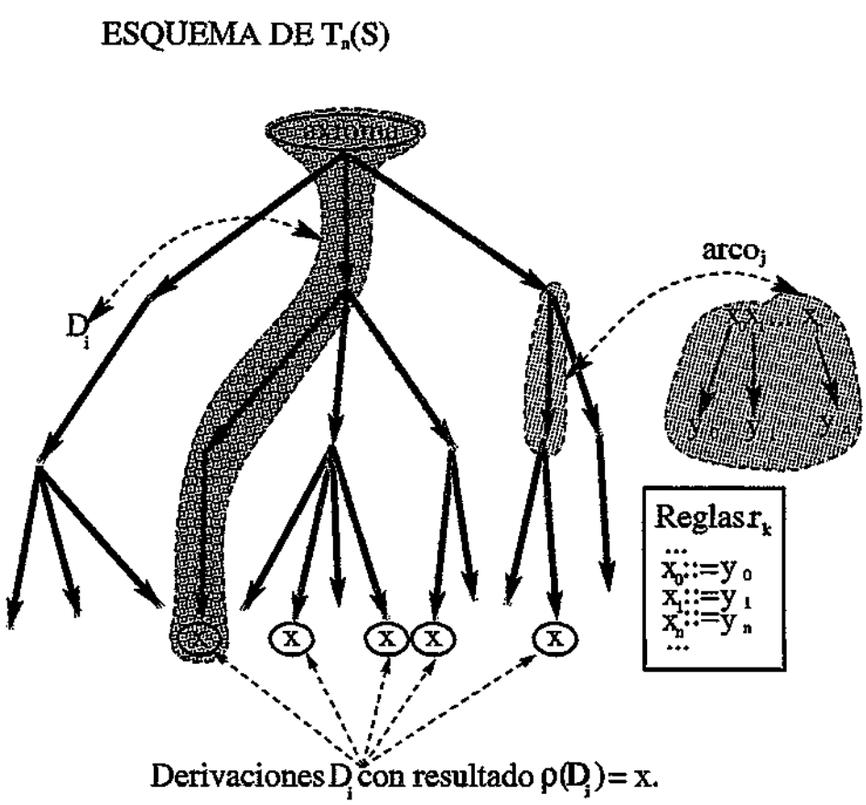


Figura 7.0.h Esquema de cálculo de probabilidades de derivar palabras.

En la figura se han resaltado:

- Las ramas cuyas hojas contienen la palabra x cuya probabilidad de ser generada está siendo calculada.

- Un arco ($arco_j$) de una de esas ramas en el que se obtiene la palabra $y_0.y_1.\cdots.y_n$ a partir de la palabra $x_0.x_1.\cdots.x_n$ utilizando para ello las reglas $r_k = x_k ::= y_k, , k \in \{0, .., n\}$.

10.17 Definición

Sea G un sistema de Lindenmayer probabilista.

Sea θ un número real.

El lenguaje generado por G y el umbral θ, que será identificado por la expresión $L(G, \theta)$, es el conjunto de cadenas que pueden ser derivadas por él a partir de su axioma y cuya probabilidad de ser generadas iguala o supera el umbral.

Formalmente,

$L(G, \theta) = \{x| \quad \exists n \in \aleph, , x \in L_n(G) \wedge p_{n,G}(x) \geq \theta\}$, donde $L_n(G)$ es el lenguaje generado por el sistema G en n pasos.

10.18 Descripción informal, equivalencia paso a paso

Dado un autómata celular probabilista, es posible describir sistemas de Lindenmayer probabilistas que sean capaces de comportarse como él. Para ello es necesario realizar una correspondencia entre palabras del sistema L y configuraciones del autómata celular. La correspondencia debe ser paso a paso, es decir, para el instante $t = 0$ es necesario que haya una palabra del lenguaje generado por el sistema L asociada a la configuración inicial del autómata. Para cada una de las configuraciones siguientes del autómata ($t = 1$) debería haber una palabra del lenguaje generada en 1 derivación asociada a la configuración del autómata, lo mismo para $t = 2$ y 2 pasos de derivación, etc...

A esta relación la llamaremos equivalendia paso a paso entre autómatas y sistemas L.

La figura 7.0.i muestra un esquema gráfico de este proceso.

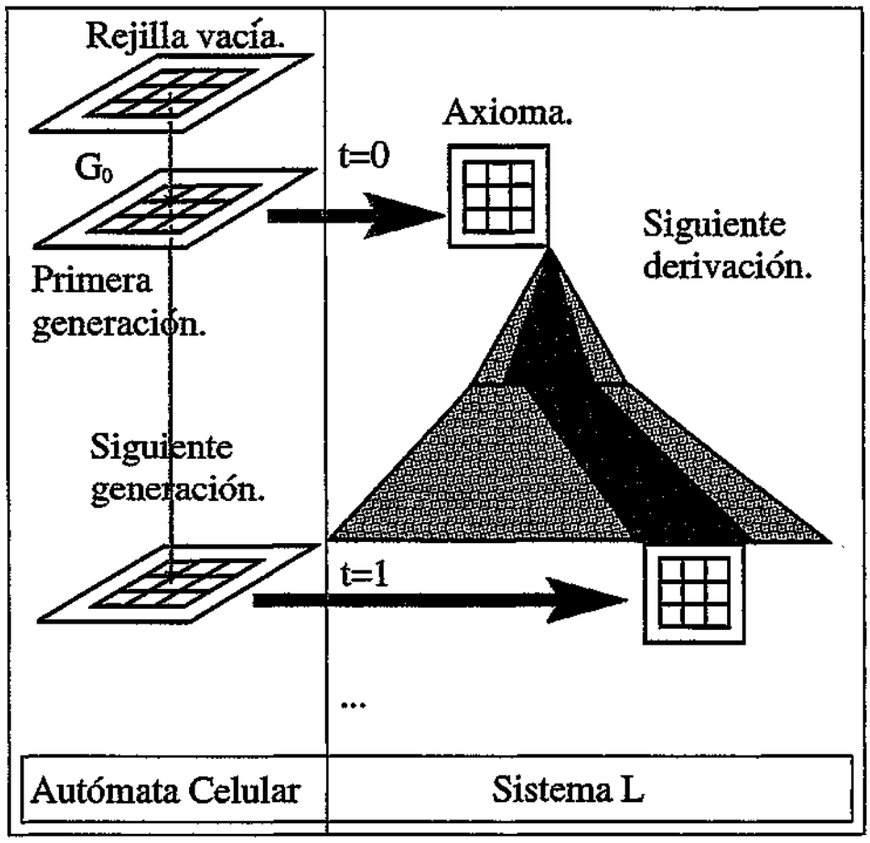


Figura 7.0.i Esquema de la equivalencia paso a paso .

10.19 Definición

Sea A un autómata celular n-dimensional probabilista.

Sea G un sistema IL n -dimensional probabilista.

Se supondrá un origen de tiempos igual a 0 y un tiempo discreto representado por números naturales.

G es equivalente paso a paso a A si y sólo si para cualquier instante de tiempo y para cualquier configuración del autómata A existe una palabra del lenguaje de G tal que la probabilidad que tiene A de estar en esa configuración en el instante considerado es la misma que la palabra tiene de ser generada por G en un número de derivaciones igual a los instantes transcurridos.

Formalmente,

$$G \text{ equivalente paso a paso a } A \Leftrightarrow \forall t \in \aleph, \forall C \text{ (configuración de A) } w \in L(G,\theta) \mid p_{t,G}(w) = p_{t,A}(C)$$

10.20 Teorema

Dado un autómata celular n-dimensional probabilista $A = (G, G_0, V, Q, F, T)$. Existe un sistema IL n-dimensional de Lindenmayer probabilista que es equivalente paso a paso a A .

Formalmente,

$\forall A = (G, G_0, V, Q, F, T)$ autómata celular n-dimensional probabilista $\Rightarrow \exists\ \langle k, 0\rangle$ sistema IL probabilista $G = \left(\sum, P^P, g, \omega, c\right) \mid A$ y G son equivalentes paso a paso.

10.21 Demostración

Consiste en una prueba constructiva, es decir, se propone un algoritmo para construir el sistema L equivalente al autómata celular de partida.

10.21.1 Determinación de $k = LongitudContexto\,(G)$

Se determina primero el tamaño del contexto del sistema L equivalente: el valor de k.

En este caso sólo hay que tener cuidado con la posibilidad de que el contexto del autómata contenga al propio autómata de la posición de la rejilla estudiada. El contexto del sistema L nunca puede contener el propio símbolo derivado. En estos casos será necesario disminuir en 1 el valor de $poblacion\,(A)$ que es el número de vecinos considerados en el autómata. Para expresar que la vecindad del autómata celular incluye al autómata de cada posición de la rejilla se recurre a la componente nula del vector de desplazamientos.

Formalmente,

$$k = \begin{cases} poblacion\,(A) & si \quad (0,...,0) \notin desplazamientos\,(A) \\ poblacion\,(A) - 1 & si \quad (0,...,0) \in desplazamientos\,(A) \end{cases}$$

10.21.2 Determinación de los símbolos del sistema L

El alfabeto del sistema L es el conjunto de estados del autómata celular, es decir:

$$\Sigma = Q$$

Además el símbolo de relleno o de marca es un símbolo nuevo, es decir

$$g \notin \Sigma$$

10.21.3 Determinación de P^P

Es un conjunto de pares $(regla, probabilidad)$ obtenido a partir del conjunto de matrices de transición del autómata de la siguiente manera:

Determinación del contexto de las reglas a partir de las configuraciones posibles de los vecindarios

Es necesario tener la misma precaución que se tuvo para determinar $LongitudContexto\,(G)$, si en la vecindad del autómata se incluye la posición del autómata estudiado $((0,...,0))$, para el contexto de las reglas debe ser excluído el símbolo del estado de esa posición.

$$P^P = \left\{ \overrightarrow{q}\, x ::= y| \quad F_{\overrightarrow{q}}\,[x,y],, \overrightarrow{q} \in Q^{poblacion(A)}, x, y \in Q \right\}$$

$$\cup \left\{ \overrightarrow{q_g} x ::= y| \quad F_{\overrightarrow{q}}\,[x,y],, \overrightarrow{q} \in Q^{poblacion(A)}, x, y \in Q \right\}$$

donde $\overrightarrow{q}$ se calcula de la siguiente forma.

$$\overrightarrow{q} = \begin{cases} \overrightarrow{q} & si\,(0,...,0) \notin desplazamientos\,(A) \\ \left(q_0, \cdots, q_{p_{0...0}-1}, q_{p_{0...0}+1}, \cdots, q_{poblacion(A)}\right) & si \begin{array}{l}(0,...,0) \in desplazamientos\,(A) \wedge \\ desplazamientos\,[p_{0...0}] = (0,...,0)\end{array} \end{cases}$$

donde se supone que el vector $\overrightarrow{q}$ puede expresarse como

$$\left(q_0, \cdots, q_{p_{0...0}-1}, q_{p_{0...0}+1}, \cdots, q_{poblacion(A)}\right)$$

y, en el caso de ser $rejilla\,(A)$ finita tiene que existir una cadena $\overrightarrow{q_g}$ para cada posible combinación de estados de vecindario para las posiciones extremas de la rejilla calculada como $\overrightarrow{q}$ pero que será completada hasta k (el k calculado al principio) con el símbolo g para las posiciones de vecindario que queden fuera de la rejilla.

10.21.4 Determinación del axioma

El axioma es la rejilla de símbolos obtenida mediante aplicación de la configuración inicial a la rejilla de autómatas del autómata celular. La elección del alfabeto Σ como el conjunto de estados Q, permite esta asociación directa.

Formalmente

$$w = G_0\,(G)$$

10.21.5 Conclusión

Es fácil ver que, por construcción, el sistema L construido es equivalente paso a paso al autómata celular de partida.

10.22 Ejemplo

10.22.1 Descripción del autómata celular

En este ejemplo se aprecia la escasa formalización habitual en la descripción de los autómatas celulares utilizados en diversas disciplinas como herramienta para estudiar sistemas dinámicos complejos. Este enunciado muestra un problema ecológico y se ha respetado la terminología y formalización propias de su área de conocimiento.

Supóngase un automata cuya evolución de campo medio sigue las ecuaciones de Lotka-Volterra para una especie de depredadores (la especie Y, que se puede considerar de carnívoros) y una especie de presas (la especie X, que se puede considerar de herbívoros) con una leve modificación que tiene en cuenta la saturación de la especie de herbívoros.

$$\begin{aligned}\frac{dN_X(t)}{dt} &= K_1 N_X(t)\left(1-\frac{N_X(t)}{N_X^{sat}}\right) - K_2 N_Y(t) N_X(t)\\ \frac{dN_Y(t)}{dt} &= -K_3 N_Y(t) + K_4 N_Y(t) N_X(t)\end{aligned}$$

donde N_X^{sat} es el grado de saturación de la especie X. El término de saturación es necesario ya que el autómata no puede representar el crecimiento ilimitado de la especie X en ausencia de individuos de la especie Y.

El territorio en el que se desarrollan estas poblaciones es una rejilla bidimensional cuadrada con condiciones de contorno periódicas para simular una rejilla infinita. Sólo se permite desplazamientos de los animales a las cuatro posiciones más cercanas del territorio.

El problema inverso, encontar las reglas de las reacciones que llevan a un conjunto específico de ecuaciones de campo medio, ha sido resuelto por Boon y otros en su extensa revisión de los autómatas de gas reactivo en retículos. Las reglas de las reacciones se codifican en una matriz de probabilidades de reacción cuyas entradas son las probailidades de obtener una configuración $\mathbf{n}^{out} = \{n_Y^{out}, n_X^{out}\}$ (número de individuos de cada una de las especies) a partir de otra $\mathbf{n}^{in} = \{n_Y^{in}, n_X^{in}\}$. En particular, una posible distribución de probabilidades que lleva a las expresiones anteriores en el límite de campo medio es la siguiente

$$\begin{aligned}p(\mathbf{n}^{in}\to\mathbf{n}^{out}) &= hK_1 n_X^{in}\delta\left(n_X^{out}, n_X^{in}+1\right)\delta\left(n_Y^{out}, n_Y^{in}\right)\left(1-\delta\left(n_X^{in}, m\right)\right)+\\ h&\left[K_2 n_X^{in} n_Y^{in} + \frac{K_1}{N_X^{sat}}\frac{m}{m-1} n_X^{in}\left(n_X^{in}-1\right) - K_1 n_X^{in}\delta\left(n_X^{in}, m\right)\right]\delta\left(n_X^{out}, n_X^{in}-1\right)\delta\left(n_Y^{out}, n_Y^{in}\right)+\\ &hK_4 n_X^{in} n_Y^{in}\delta\left(n_X^{out}, n_X^{in}\right)\delta\left(n_Y^{out}, n_Y^{in}+1\right)\left(1-\delta\left(n_Y^{in}, m\right)\right)+\\ h&\left[K_3 n_Y^{in} - K_4 n_X^{in} n_Y^{in}\delta\left(n_Y^{in}, m\right)\right]\delta\left(n_X^{out}, n_X^{in}\right)\delta\left(n_Y^{out}, n_Y^{in}-1\right) \quad \text{for } \mathbf{n}^{in}\neq\mathbf{n}^{out}\\ p(\mathbf{n}^{in}\to\mathbf{n}^{in}) &= 1-\sum_{\mathbf{n}^{out}\neq\mathbf{n}^{in}} p(\mathbf{n}^{in}\to\mathbf{n}^{out})\end{aligned} \tag{10.1}$$

donde $\delta(n,n')$ es una delta de Kronecker (un indicador igual a 1 si $n = n'$ y 0 en otro caso), el inverso de h representa la escala de tiempo de reacción y m es el máximo número de individuos de una especie en un momento determinado que coincide con el número de canales asociados con cada nodo. En el modelo que se muestra $m = 4$, que quiere decir que 4 es el máximo número de individuos de cada especie que pueden ocupar una cuadrícula de territorio (un nodo del autómata).

La condición para que $p(\mathbf{n}^{in}\to\mathbf{n}^{out})$ sea una probabilidad, es decir, un número no negativo en el intervalo $[0,1]$, impone algunas restricciones a los posibles valores de las constantes de la reacción K_i , h y N_X^{sat} que se pueden utilizar en estas simulaciones. En particular N_X^{sat} debería ser menor o igual que m (esta cota superior corresponde a la plena ocupación de un nodo del autómata).

10.22.2 Construcción del sistema IL equivalente paso a paso.

Cada autómta de la rejilla contiene varios individuos de cada especie. Llamaremos $l = x(t)$ y $k = y(t)$ al número de individuos de la especie x e y en el instante t. La distribución de probabilidades descrita anteriormente puede representarse por medio del diagrama de estados de la figura 7.0.j donde

$$p_1 = \left\{ \begin{array}{ll} K_1 x(t) & x(t) \neq m \\ 0 & x(t) = m \end{array} \right\}, p_2 = \left\{ \begin{array}{ll} K_4 x(t) y(t) & y(t) \neq m \\ 0 & y(t) = m \end{array} \right\},$$

$$p_3 = max\left(0, \left\{ \begin{array}{ll} K_3 y(t) & y(t) \neq m \\ K_3 y(t) - K_4 x(t) y(t) & y(t) = m \end{array} \right\}\right),$$

$$p_4 = max\left(0, \left\{ \begin{array}{ll} K_2 x(t) y(t) + \frac{K_1}{N_X^{sat}} \frac{m}{m-1} x(t)\left[x(t) - 1\right] & x(t) \neq m \\ K_2 x(t) y(t) + \frac{K_1}{N_X^{sat}} \frac{m}{m-1} x(t)\left[x(t) - 1\right] - K_1 x(t) & x(t) = m \end{array} \right\}\right)$$

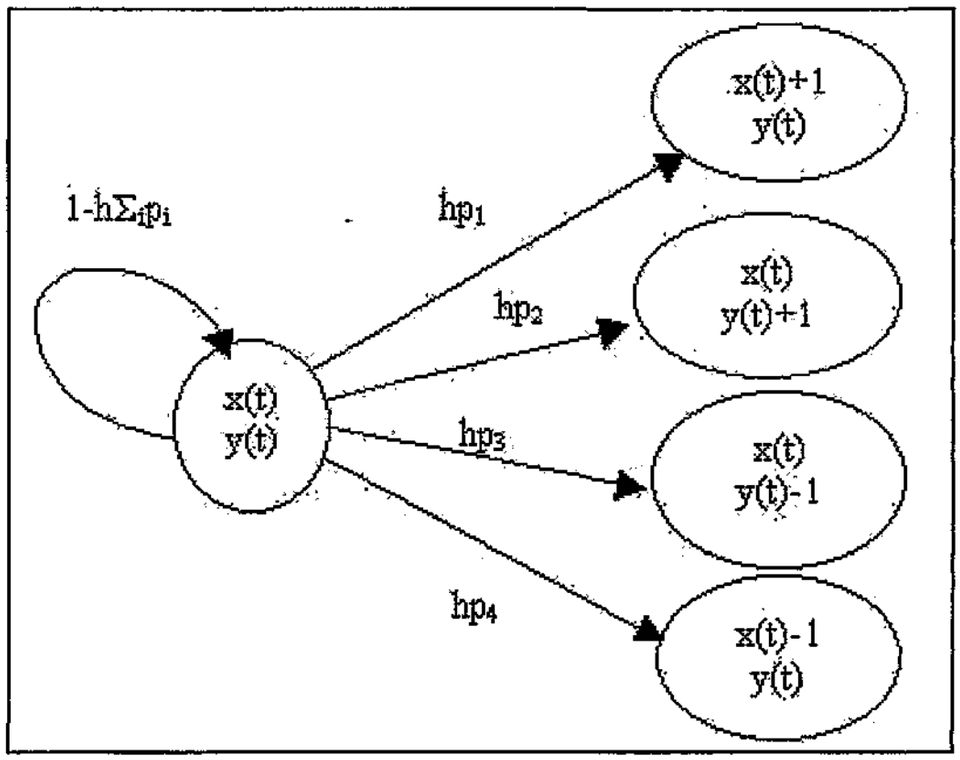


Figura 7.0.j Diagrama de estados.

El sistema equivalente no necesita interacciones.

Su alfabeto es el conjunto de pares cada una de cuyas componentes es el número de posibles individuos de cada especie:

$$\Sigma_P = \{(l,k) \quad \forall l,k \in \{0,1,2,3,4\}\}$$

El conjunto de reglas de producción puede obtenerse de la siguiente tabla donde el símbolo

de la parte izquierda de cada regla tiene como subíndice t y el de la parte derecha $t+1$.

$$P_P = \{((l_t, k_t) ::= (l_{t+1}, k_{t+1}), M[(l_t, k_t), (l_{t+1}, k_{t+1})])\}$$

Donde M es la matriz que se muestra a continuación (se está utilizando (l_t, k_t) como índice por filas y (l_{t+1}, k_{t+1}) como índice por columnas). $M[(l_t, k_t), (l_{t+1}, k_{t+1})]$ es la probabilidad contenida en cada casilla aunque la tabla también tiene el símbolo explícito de la parte derecha de la regla que es información redundante porque se puede obtener de las entradas a la tabla.

$(l,k)_t$	$(l,k)_{t+1}$	$(l+1,k)_{t+1}$	$(l,k+1)_{t+1}$	$(l,k-1)_{t+1}$	$(l-1,k)_{t+1}$
(0,0)	(0,0),1	-	-	-	-
(0,1)	(0,1),1	-	-	-	-
(0,2)	(0,2),1	-	-	-	-
(0,3)	(0,3),1	-	-	-	-
(0,4)	(0,4),1	-	-	-	-
(1,0)	(1,0),1	-	-	-	-
(1,1)	$(1,1), 1-\Sigma p_i$	$(2,1), hK_1$	$(1,2), hK_4$	$(1,0), hK_3$	$(0,1), hp_4$
(1,2)	$(1,2), 1-\Sigma p_i$	$(2,1), hK_1$	$(1,3), 2hK_4$	$(1,2), 2hK_3$	$(0,2), hp_4$
(1,3)	$(1,3), 1-\Sigma p_i$	$(2,3), hK_1$	$(1,4), 3hK_4$	$(1,2), 3hK_3$	$(0,3), hp_4$
(1,4)	$(1,4), 1-\Sigma p_i$	$(2,4), hK_1$	-	$(1,3), 4h(K_3-K_4)$	$(0,4), hp_4$
(2,0)	(2,0),1	-	-	-	-
(2,1)	$(2,1), 1-\Sigma p_i$	$(3,1), 2hK_1$	$(2,2), 2hK_4$	$(2,0), hK_3$	$(1,1), hp_4$
(2,2)	$(2,2), 1-\Sigma p_i$	$(3,2), 2hK_1$	$(2,3), 4hK_4$	$(2,1), 2hK_3$	$(1,2), hp_4$
(2,3)	$(2,3), 1-\Sigma p_i$	$(3,3), 2hK_1$	$(2,4), 6hK_4$	$(2,2), 3hK_3$	$(1,3), hp_4$
(2,4)	$(2,4), 1-\Sigma p_i$	$(3,4), 2hK_1$	-	$(2,3), 4h(K_3-2K_4)$	$(1,4), hp_4$
(3,0)	(3,0),1	-	-	-	-
(3,1)	$(3,1), 1-\Sigma p_i$	$(4,1), 3hK_1$	$(3,2), 3hK_4$	$(3,0), hK_3$	$(2,1), hp_4$
(3,2)	$(3,2), 1-\Sigma p_i$	$(4,2), 3hK_1$	$(3,3), 6hK_4$	$(3,1), 2hK_3$	$(2,2), hp_4$
(3,3)	$(3,3), 1-\Sigma p_i$	$(4,3), 3hK_1$	$(3,4), 9hK_4$	$(3,2), 3hK_3$	$(2,3), hp_4$
(3,4)	$(3,4), 1-\Sigma p_i$	$(4,4), 3hK_1$	-	$(3,3), 4h(K_3-3K_4)$	$(2,4), hp_4$
(4,0)	(4,0),1	-	-	-	-
(4,1)	$(4,1), 1-\Sigma p_i$	-	$(4,2), 4hK_4$	$(4,0), hK_3$	$(3,1), hp_4$
(4,2)	$(4,2), 1-\Sigma p_i$	-	$(4,3), 8hK_4$	$(4,1), 2hK_3$	$(3,2), hp_4$
(4,3)	$(4,3), 1-\Sigma p_i$	-	$(4,4), 12hK_4$	$(4,2), 3hK_3$	$(3,3), hp_4$
(4,4)	$(4,4), 1-\Sigma p_i$	-	-	$(4,3), 4h(K_3-4K_4)$	$(3,4), hp_4$

El axioma w_P es una matriz aleatoria de elementos de Σ_P.

El sistema completo es (Σ_P, P_P, w_P)

Parte IV

Conclusiones y líneas abiertas

Sistemas L y fractales

Conclusiones

La presente tesis engloba y completa aspectos presentes en publicaciones previas [Alf95], [Alf96] y [Alf97b] Nuestra comparación entre fractales y sistemas L tiene como objetivo comprobar la potencia expresiva de estos últimos.

Es conocido que los objetos fractales son capaces de expresar conductas complejas. También es conocido que algunos fractales se pueden representar mediante una combinación compuesta por un sistema L y una interpretación gráfica de sus símbolos. Por lo tanto, que los sistemas L son lo suficientemente expresivos como para representar algunos fractales, es un hecho conocido. Se plantea, entonces, la posibilidad de analizar propiedades de los fractales a través de los sistemas L que los representan en lugar de directamente sobre las gráficas.

El hecho de que los trabajos realizados hasta el momento reflejen un análisis más bien gráfico y la incorporación de las interpretaciones gráficas de las cadenas como un paso obligado para la representación de fractales con sistemas L, sugiere la conveniencia de un análisis más formal de las aportaciones de la interpretación gráfica al proceso.

En este sentido se ha demostrado que las dos interpretaciones gráficas utilizadas en la literatura (interpretación gráfica de tortuga e interpretación gráfica vectorial) junto con los sistemas L que las utilizan son equivalentes siempre que se cumplan algunas condiciones:

Los sistemas L que utilizan la interpretación gráfica de tortuga son invariantes al ángulo, es decir, la interpretación gráfica de sus cacenas deja la tortuga apuntando a la misma dirección y sentido que tenía inicialmente.

Los sistemas L que utilizan la interpretación gráfica vectorial están relacionados racionalmente, es decir, tanto para los ángulos como para los módulos de los vectores asociados a sus símbolos existe un valor mínimo (ángulo y módulo elementales) y todos los demás pueden ser expresados como múltiplos naturales de estos valores elementales.

El cumplimiento de estas restricciones supone una pérdida de generalidad mínima. La mayoría de los sistemas L utilizados para representar fractales en la literatura son invariantes al ángulo o están racionalmente relacionados. Además,en casos en los que los sistemas L no cumplían las condiciones se ha podido encontrar otros sistemas que generan la misma gráfica y que sí las satisfacen.

Respecto al estudio de propiedades de los fractales a través de los sistemas L que los repre-

sentan hemos definido dos dimensiones para un subconjunto de fractales que, en la mayoría de los casos, devuelven los mismos valores que las otras definiciones de dimensión. La nuestra es el límite de un cociente en el que tanto el numerador como el denominador pueden ser calculados analizando exclusivamente las cadenas generadas por los sistemas de Lindenmayer.

El subconjunto de fractales estudiado incluye aquellos que pueden ser representados mediante algún sistema L cuyas reglas tengan todas la misma estructura.

Las dos dimensiones definidas utilizan la longitud recorrida por la curva en su trazo por el plano $\Re^2$, realizado sin levantar el lápiz del papel. En muchos casos este trazo pasa más de una vez por algunos tramos de la curva. Ésta es la razón que aconsejó introducir dos definiciones de dimensión. Una de ellas no tiene en cuenta que algunos tramos son recorridos más de una vez mientras que la otra elimina los solapamientos.

Para resolver las dificultades asociadas al proceso de eliminación fue necesario representar los puntos del plano con una notación basada en tuplas de números enteros en lugar de pares de números reales. La forma canónica de esta representación permitía calcular correctamente la longitud deseada.

Otro resultado que se muestra es que la primera de las dimensiones es una cota superior de la segunda.

Como consecuencia de los párrafos anteriores podemos concluir:

- Que la potencia expresiva de los sistemas L parece adecuada para la generación de modelos de sistemas complejos, por ejemplo los fractales de tipo *iniciador-iterador*.
- Que entre sistemas L y fractales se realiza un cambio de dominio, gracias a interpretaciones gráficas adecuadas.
- Que expresar los fractales mediante sistemas L permite utilizar técnicas propias de éstos (estudio de las cadenas derivadas) en la resolución de problemas de aquéllos (dimensión).

Futuras líneas de trabajo

Los resultados obtenidos se han limitado a algunos conjuntos de sistemas L y de fractales y al estudio de la dimensión de éstos.

En el futuro nos proponemos

- Ampliar los estudios a otros fractales de tipo distinto al *iniciador-iterador* (fractales autoafines, fractales de tipo conjunto de Julia o Mandelbrot biomorfos, movimiento browniano).
- Dentro del tipo de fractales *iniciador-iterador*, ampliar el estudio a sistemas L con interpretación gráfica tortuga que no sean invariantes al ángulo y a sistemas L con interpretación gráfica vectorial que no estén relacionados racionalmente.
- Estudiar, a partir de los sistemas L equivalentes, otras características de los fractales distintas de la dimensión.

- Dentro del estudio de la dimensión, ampliarlo a fractales que no se puedan representar con sistemas L de reglas con la misma estructura.

Sistemas L y autómatas celulares

Conclusiones

El objetivo de la comparación entre sistemas L y autómatas celulares es comprobar la potencia expresiva de estos últimos. El resultado obtenido es que, al menos en todos los tipos de autómatas celulares estudiados, se ha podido obtener sistemas L que se comportan como ellos.

Se ha elegido un autómata celular de cada una de las tres dimensiones que tienen representación gráfica.

El unidimensional es uno de los catalogados por Wolfram [Wol94] dentro de los que tienen tres vecinos: los dos autómatas finitos más próximos y el estudiado.

El bidimensional simula el comportamiento de un ecosistema con dos especies: presa y depredador. Los depredadores pueden estar a su vez en dos estados. Los individuos de la misma región pueden depredar y reproducirse. Está permitido que los individuos se desplacen por todo el territorio.

El tercero es un autómata tridimensional que simula la generación y propagación de un pulso por el eje central de un prisma cuadrado recto semi infinito por uno de sus extremos.

La mayor dificultad en el estudio de estos autómatas consiste en la escasa formalización de su enunciado. De hecho, una vez formalizados, la consecución del sistema L equivalente resulta sencilla. Es difícil proponer un método general para formalizar autómatas celulares descritos de manera informal. Por lo tanto, es difícil proponer un método general para obtener el sistema L que se comporte igual que un autómata celular expresado de manera informal.

Para comprobar que realmente la dificultad en la consecución de sistemas L equivalentes radica en la falta de formalización de los autómatas celulares se obtuvo un resultado más general: se ha demostrado que para todo autómata n-dimensional probabilista (aquellos no deterministas que asignan probabilidades a cada regla de producción) se puede construir un sistema L equivalente. La posibilidad de expresar otros tipos de autómatas como caso particular de los probabilistas permite utilizar el resultado general en los particulares.

Se obtuvo otro resultado de índole práctico. Durante el desarrollo de esta tesis se escribieron programas para implementar tanto las operaciones de sistemas L como las de autómatas celulares. Cuando se comparó la eficiencia de los programas para obtener las cadenas derivadas por el sistema L y las generaciones de los autómatas, se comprobó que, en algunos caos, las primeras resultaban sensiblemente más eficientes.

Todo lo expuesto anteriormente nos permite concluir:

- Que los sistemas L son al menos igual de expresivos que los autómatas celulares n-dimensionales.
- Que el principal problema para tratar con autómatas celulares consiste en la poca formalización con la que son presentados.
- Que la expresión del autómata celular mediante un sistema L permite utilizar técnicas propias de éstos con resultados a veces más eficientes.

Sistemas L y autómatas celulares: futuras líneas de trabajo

El principal objetivo para el futuro en este tema es formalizar las relaciones entre sistemas L y autómatas celulares de forma que se pueda estudiar las propiedades de los autómatas a través del sistema L que se comporte igual.

Otro objetivo es extender el resultado obtenido para los autómatas celulares n-dimensionales probabilistas a los autómatas celulares n-dimensionales deterministas y a los autómatas celulares n-dimensionales no deterministas sin probabilidades mediante la construcción de autómatas n-dimensionales probabilistas equivalentes.

Otras líneas futuras.

Se pretende seguir estudiando las propiedades formales de los sistemas L.

Para ello se continuará comprobando su poder expresivo comparándolos con otros dominios y formalismos.

Desde un punto de vista práctico, consideramos interesante profundizar en la aplicación, tanto de los sistemas L como de los autómtas celulares, a la simulación. Nos planteamos integrar estos dos formalismos en herramientas de simulación, para poder ofrecer en el futuro nuevas vías para el estudio de sistemas complejos a otras disciplinas científicas.

Parte V

Referencias

Clave *Referencia*

[*Alf*95] Alfonseca, M., Ortega, A.,
Fractales gramaticales.
Investigación y Ciencia. Febrero 1995. 221

[*Alf*96] Alfonseca, M., Ortega, A.,
Representation of fractal curves by means of L Systems.
APL96 International Conference on APL

[*Alf*97*a*] Alfonseca, M., Sancho, J. y Martínez, M., 1997
Teoría de Lenguajes, Gramáticas y Autómatas. Textos de Cátedra.
Publicaciones R.A.E.C.

[*Alf*97*b*] Alfonseca, M., Ortega, A.
A study of the representtion of fractal curves by L systems and their equivalences.
IBM Journal of Research and Development. vol. 41, 6. november 97

[*Alf*98] Alfonseca, M., Lara, J. de, Pulido, E., 1998
Educational simulation of complex ecosystems in the World-Wide Web.
Proc. ESS'98, SCS Int. pp. 248-252

[*Ans*93] Anson, L. F., 1993
Fractal Image Compression. Byte. October 1993

[*Bar*88] Barnsley, M., 1988
Fractals Everywhere
Academic Press, San Diego

[*Bed*94] Bedford, T., Dekking, F.M., Breeuwer, M., Keane, M.S., van Schooneveld, D.,
Fractal Coding of Mononchrome Images.
Signal Procesing: Image Communication, vol 6, pp. 405-419

[*Boo*96] Boon, J. P., Dab, D., Kapral, R., Lawnizcak, A.
Lattice Gas Automata for Reactive Systems.
Physics Reports 273, pp.55-147, 1996

[*Brs*92] Boers, E., Kuiper, H., 1992
Biological metaphors and the design of modular artificial networks.
Master's thesis. Departments of Computer Science and Experimental and Theoretical Psychology at Leiden University, the Netherladns

[*Buj*89] Bujalance, E., Etayo, J. J., Gamboa, J. M., 1989
Teoría Elemental de Grupos. Cuadernos de la UNED.
Mateu Cromo Artes Gráficas, S.A.

Clave *Referencia*

[*Bur*70] Burks, A. W., 1970
Essays on cellular automata. University of Illinois Press.

[*Can*86] Canavos, G. C., 1986
Probabilidad y Estadística. Aplicaciones y métodos. McGraw Hill

[*Cas*94] Casey, S. D., Reignold, N. F.,
Self-Similar Fractal Sets: Theory and Procedure. IEEE Computer Graph & Appl. 14 73-82 (May 1994)

[*Cas*94] Castellet, M., Llerena, I., 1994
Álgebra lineal y geometría. Editorial Reverté, S.A. Universidad Autónoma de Barcelona.

[*Cnw*] Conway, J.H., Berlekamp, E. R., Guy, R. K.,
Winning ways for your mathematical plays.
New York: Academic Press, Vol 2, Cap. 25

[*Cor*93] Corbit, J. D., Garbary, D. J., 1993
Computer simulation of the morphology and development of several species of seaweed using Lindenmayer systems.
Computers & Graphics (Jan.-Feb. 1993) vol. 17, 1 pp.85-8

[*Cos*91] Costa González, A. F., Lafuente López, J. 1991
Geometrías Lineales y grupos de transformaciones. Cuadernos de la UNED.
Simancas Ediciones, S.A.

[*Cul*91*a*] Culik II, K., and Dube, S., 1991
New Methods for Image Generation and Compression
New Results and New Trends in Computer Science. Proceedings.
Maurer, H. Eds. Berlin, Germany: Springer-Verlag, 1991 pp. 69-90

[*Cul*91*b*] Culik II, K., and Dube, S., 1991
Balancing order and chaos in image generation.
Automata, Languages and Programming. 18th International Colloquium Proceedings. Albert, J.L., Monien, B., Artalejo, M.R.
Berlin, Germany: Springer-Verlag, 1991 pp.600-614

[*Cul*92] Culik II, K., and Dube, S., 1992
L-System and mutually recursive function systems.
Acta Informática 1993 vol. 30, 3 pp.279-302

Clave	*Referencia*
[*Dek*82*a*]	Dekking, F. M., 1982 Recurrent Sets. Advances in Mathematics. vol. 44:1, pp 78-104
[*Dek*82*b*]	Dekking, F. M., 1982 Recurrent Sets: a fractal formalism. Technical report 82-32. Technische Hogeschool, Delft.
[*Des*1637]	Descartes, R., 1637 La geometría.
[*Euc*]	Euclides Elementos
[*Fal*90]	Falconer, K., Fractal Geometry: Mathematical Foundations and Applilcations. John Wiley & Sons.
[*Flk*98]	Flake, G., 1998 The Computational Beauty of Nature. Computer explorations of fractals, chaos, complex systems, and adaptation. The M.I.T. Press
[*Gie*91]	Giessmann, E. G., 1991 Generation of fractal curves by generalizations of Lindenmayer's L Systems. Proceedings of the 1st IFIP Conference on Fractals in the Fundamental and Applied Sciences. H.-O. Peitgen, J. M. Henriques, y L. F. Penedo, Eds., North-Holland, Amsterdam, 1991, pp. 147-157
[*Har*92]	Hart, J. C., 1992 The object instancing paradigm for linear fractal modeling. Porceedings Graphics Interface '92. Toronto, Ont., Canada: Canadian Inf. Process. Soc, 1992, pp. 224-31. Morgan Kaufmann Publishers.
[*Hil a*]	Hilbert, D., Fundamentos de la Geometría.
[*Hil b*]	Hilbert, D., Cohn-Vossen, S. Geometry and the imagination.
[*Jür*90]	Jürgens, H., Peitgen, H-O., Saupe, D., 1990 El lenguaje de los fractales. Investigación y Ciencia, noviembre de 1987.

Clave	*Referencia*
[*Kaa*94]	Kaandorp, J. A., 1994 Fractal Modelling Grotwth and Form in Biology. Springer-Verlag.
[*Kar*95]	Kari, J., 1995 Cellular Automata. An Introduction, in Artificial Life: Grammatical Models. G. Paun Eds. Black Sea Univ. Press., Bucharest, 1995
[*Koz*93]	Koza, J. R., 1993 Discovery of Rewrite Rules in Lindenmayer Systems and State Transition Rules in Cellular Automata via Genetic Programming. Symposium on Pattern Formation (SPF-93)
[*Lin*75]	Lindenmayer, A., Rozenberg, G. y Herman, G., 1975 Developmental Systems and Languages. North-Holland/American Elservier
[*Lin*90]	Lindenmayer, A., Prusinkiewicz, P., 1990 The algorithmic beauty of plants. Springer-Verlag.
[*Man*77]	Mandelbrot, B., 1977 La Geometría Fractal de la Naturaleza. Tusquets Editores
[*Miy*89]	Miyata, K., 1989 A Method of Generating Cloud Images Using Density Contour Lines. The Transactions of the IEICE, vol. e 72, 6. june 1989
[*Miy*90]	Miyata, K., 1990 A Method of Generating Stone Wall Patterns. Computer Graphics, vol. 24, 4, August 1990
[*Neu*66]	von Neumann, J., Theory of Self-Reproducing Automata. University of Illinois Press. Urbana
[*Pap*80]	Papert, S., 1980 Mindstorms:Children, Computers, and Powerful Ideas. Basic Books, New York, 1980
[*PeiRic*]	Peitgen, H. O., Richter, P. H., The Beauty of Fractals. Springer-Verlag.

Clave *Referencia*

[*Pru*86] Prusinkiewicz, P., 1986
Graphical Applications of L-Systems.
Proceedings of Graphical Interface 86 and Vision Interface 86.
M. Wein and E. M. Kidd, Eds. Vancouver, BC, 1986 pp. 247-253

[*Pru*90] Prusinkiewicz, P., Lindenmayer, A., Fracchia, F.D.,1990
Synthesis of space-filling curves on the square grid.
Fractals in the Fundamental and Applied Sciences. Proceedings of the First IFIP Conference.
Peitgen, H.-O., Henriques, J.M., Penede, L.F., Eds. Amsterdam, Netherlands North-Holland, 1991, pp.341-66

[*Que*79] Queysanne, M., 1979
Álgebra Básica. Editorial vicens-vives.

[*Roz*92] Rozenberg, G., Salomaa, A. (Eds.)
Lindenmayer Systems. Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology.
Springer-Verlag, Berlin 1992.

[*San*87] Sander, L. M., 1987
Crecimiento fractal. Investigación y Ciencia, marzo de 1987.

[*Sip*97] Sipper, M., Mange, D., and Stauffer, A., 1997
Ontogenetic hardware. Biosystems 44, pp. 193-207

[*Sip*98*a*] Sipper, M., Stauffer, A., 1998
On the relationship between cellular automata and L-Systems: The self-replicant case. Physica D 116, pp.71-80

[*Sip*98*b*] Sipper, M., Stauffer, A., 1998
L-hardware: Modeling and implementing cellular development using L-Systems. In D. Mange and M. Tomassini, editors,
Bio-inspired Computing Machines: Toward Novel Computational Architectures. Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, pp. 269-287

[*Smi*91] Smith, H. F., 1991
A garden of fractals.
Fractals in the Fundamental and Applied Sciences. Proceedings of the first IFIP Conference. Peitgen, H.-O., Henriques, J. M., Penedo, L. F., Eds.
Amsterdam, Netherlands: North-Holland pp. 407-24.

Clave	*Referencia*
[*Smt*71]	Smith, A. R., 1971 Simple computation-universal cellular spaces. J. ACM. 18, 331
[*Tru*]	TruSoft Int'l Inc. Help file from Benoît application.
[*Vol*31]	Volterra, V. 1931 Leçons sur la Théorie Mathématique de la Lutte pour la Vie. Gauthier - Villars, Paris, 1931
[*Wei*91]	Weisbuch, G., 1991 Complex systems dynamics. A lecture notes volume in the Santa Fe Institute studies in the sciences of complexity. Addison-Wesley Publishing Company.
[*Wol*94]	Wolfram, S., 1994 Cellular Automata and Complexity. Collected papers. Addison-Wesley Publishing Company
[*Yam*93]	Yamaguti, M., Hata, M., Kigami, J. 1993 Mathematics of Fractals. American Mathematical Society

Reunido el tribunal que suscribe en el día de la fecha, acordó calificar la presente Tesis doctoral con SOBRESALIENTE CUM LAUDE P.U.
Madrid, 23 DE JUNIO DE 2000

Presidente
Luis de Ledesma

VOCAL 2
Blanca Rhus

LO SECRETARIO.
FDO. PILAR RODRÍGUEZ

Vocal 1
M.

VOCAL 3
FDO. PABLO CASTELLS
Pablo Castells