INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Diseño e implementación de un modelo que describa la dinámica de un autómata celular utilizando redes de Petri

Tesis
que para obtener el grado de
Maestro en Ciencias de la Computación

Presenta
Mario Martínez Molina

Director de Tesis: M. en C. Germán Téllez Castillo

México, D. F. 22/06/2010

Agradecimientos

A Dios por permitirme apreciar la belleza del universo.

A mi familia por su incondicional cariño y apoyo, sin los cuales jamás habría podido realizar este sueño.

Al M. en C. Germán Téllez Castillo por haber sido un guía incansable durante el desarrollo de este trabajo.

A todas aquellas personas que han dedicado su vida al sublime propósito de desarrollar la ciencia.

Al ingeniero Carlos Ignacio Reséndiz Juárez y al M. en C. Alejandro León Javier por su amistad durante los momentos más difíciles de mi vida.

Al Centro de Investigación en Computación por la formación y vocación de investigador que me ha dado.

Al CONACYT por su apoyo continúo a los estudiantes mexicanos.

A mi alma mater el Instituto Politécnico Nacional por haber hecho de mí una mejor persona.

Tabla de contenido

Índice de figuras

Índice de tablas

Resumen

La necesidad de modelar las interacciones entre individuos y su ambiente es un problema fundamental en la ecología. No obstante la complejidad de los ecosistemas, es posible alcanzar un entendimiento básico de los fenómenos más representativos, de modo que, sin dejar de lado esta enorme complejidad, es posible encontrar patrones recurrentes que faciliten el análisis de las interacciones que se dan entre los individuos que los habitan.

En este trabajo de tesis se presenta la modelación de la dinámica de la competencia y depredación presente en un ecosistema de depredación secuencial, a través de un modelo basado en autómatas celulares. La dinámica de la competencia es descrita a detalle, no solo desde el enfoque ecológico, sino también del de los modelos espaciales. En este sentido, fenómenos como la exclusión, coexistencia y el antagonismo mutuo son descritos considerando las interacciones locales presentes no solo en los ecosistemas, sino también en los autómatas celulares.

Dada la importancia de la dinámica de las interacciones locales en el comportamiento macroscópico de un sistema dinámico, se ha recurrido a las redes de Petri para modelar estas interacciones. Las redes de Petri son una técnica de modelado formal, con cambios de estados asíncronos, concurrentes y / o basados en eventos, donde se utilizan símbolos gráficos para la descripción de estos cambios de estado. Esto permite la descripción formal del comportamiento de un autómata celular dentro de una fase de tiempo síncrona que representa una etapa de evolución del sistema.

Los resultados muestran que el modelo propuesto no solo es capaz de reproducir la dinámica del ecosistema bajo estudio sino que, a diferencia de los modelos ecológicos clásicos basados en ecuaciones diferenciales y ecuaciones en diferencias, produce patrones complejos que solo son observables en modelos espaciales como los autómatas celulares y los “lattice gas”. Estos patrones son discutidos y situados dentro de un contexto ecológico, en el cual los individuos tienen un papel primordial como fuerza motora de un ecosistema.

Abstract

The need to model the interactions between individuals in an ecosystem is a fundamental problem in ecology. Nevertheless the inherent complexity of an ecosystem, it is possible to achieve a basic understanding of the most representative phenomena, in such a way that, while recognizing this complexity, recurrent patterns can be found in order to facilitate the analysis of the interactions between the individuals that inhabit these ecosystems.

In this dissertation a cellular automata based model of the dynamics of competition and predation, in an ecosystem subjected to sequential predations is presented. The dynamics of competition is described thoroughly, not only from an ecological viewpoint, but from the perspective of spatial models as well. According to this, phenomena such as exclusion, coexistence and mutual antagonism are described in terms of the local interactions present not only in the ecosystems, but in cellular automata as well.

Since local interactions are of utmost importance for the resulting behavior of dynamical systems, Petri nets are used to model such interactions. Petri nets are a formal modeling technique with event based, concurrent or asynchronous state changes, where graphical symbols are used for the description of state changes. This allows the formal description of a cellular automaton's behavior within a synchronous time stage that represents one evolution phase of the system.

The results show that the proposed model not only exhibits the desired dynamics of the ecosystem under study, but in contrast to the classic ecological models of differential and difference equations, it produces patterns that can only be observed in spatial models. Such patters are discussed and placed in an ecological context, where individuals take a primordial role as a driving force of an ecosystem.

Capítulo I. Introducción

1.1. Planteamiento del problema

La necesidad de modelar las interacciones entre individuos y su ambiente es un problema fundamental en la ecología. Esta problemática se acentúa debido a la falta de unicidad en los ecosistemas: *existen millones de diferentes especies, así como millones de organismos genéticamente distintos que viven e interactúan en un mundo siempre cambiante* [1]. No obstante, a través del estudio de los ecosistemas, es posible alcanzar un entendimiento básico de los fenómenos más representativos, de modo que, sin dejar de lado la enorme complejidad de los ecosistemas, es posible encontrar patrones recurrentes que faciliten el análisis de las interacciones que se dan entre los individuos que los habitan.

Los autómatas celulares (CA) han sido utilizados extensamente para describir sistemas dinámicos, cuyo comportamiento macroscópico es resultado de la "suma" total de las interacciones locales entre los "elementos" que componen dichos sistemas. Este enfoque contrasta con los modelos clásicos, donde es común introducir parámetros dependientes del tiempo y espacio (e. g. el índice de refracción), para explicar cierto comportamiento, que a nivel microscópico depende de las interacciones antes mencionadas.

Al igual que el estudio de las interacciones entre los individuos que habitan un ecosistema ayuda a su comprensión y estudio; el estudio y modelado de las interacciones locales de un AC facilitan la comprensión y estudio del esquema de cómputo presente en los AC.

Las redes de Petri son una técnica de modelado formal basada en eventos, con cambios de estado asíncronos o concurrentes, donde se utilizan símbolos gráficos para la descripción de estos cambios de estado. Esto permite la descripción formal del comportamiento de un autómata celular dentro de una fase de tiempo síncrona que representa una etapa de evolución del sistema. Esto puede ser utilizado para modelar el movimiento de individuos que atraviesan un número arbitrario de células en una fase de tiempo. Dado que las Redes de Petri permiten una representación gráfica, es posible visualizar el comportamiento de las células modeladas, así como sus interacciones y cadenas de causa y efecto.

1.2. Objetivos

1.2.1. Objetivo general

Modelar un ecosistema de depredación secuencial usando un modelo basado en autómatas celulares, y describir las interacciones locales del mismo a través de una red de Petri.

1.2.2. Objetivos particulares

a) Realizar la investigación necesaria sobre sobre autómatas celulares (clásicos y de bloque), redes de Petri y autómatas probabilísticos para resolver el problema bajo estudio.
b) Modelar las interacciones locales de un autómata celular a través de una Red de Petri.
c) Modelar un ecosistema bajo depredación secuencial a través de un autómata celular.
d) Discutir y analizar los patrones espaciales generados por un autómata celular.
e) Determinar la importancia de las interacciones entre los individuos que componen un ecosistema de depredación secuencial.
f) Determinar las condiciones necesarias para reproducir el fenómeno de “Herbivory in minority species”.
g) Implementar un sistema de software en la plataforma Microsoft .Net, usando el lenguaje de programación C#, de manera tal que sea posible simular el modelo propuesto.

1.3. Justificación

Los modelos de sistemas ecológicos orientados a individuos, están basados en la descripción detallada del comportamiento de cada individuo y sus interacciones con los demás. Todos los individuos de un modelo pueden tener atributos que varían con el tiempo como: la edad, el peso, una posición en un área, etc. Además cada uno de estos atributos es capaz de influenciar el comportamiento de los demás individuos del modelo. En un modelo típico los individuos nacen o mueren de acuerdo a circunstancias específicas que los rodean en un instante dado de tiempo. Por lo que la conducta del modelo en su totalidad es descrita agregando el comportamiento de cada individuo así como sus interacciones.

Una ventaja importante de esta aproximación orientada a individuos, es el nivel de detalle en que permite modelar las interacciones complejas de un sistema. Modelos globales basados en poblaciones o clases de individuos como, por ejemplo, modelos de $p - estados$, no contienen información si los individuos de una población son vecinos unos de otros. Los modelos orientados a individuos, en su lugar permiten interacciones locales entre individuos. Esto es importante especialmente para la modelación de sistemas con distribución no homogénea de individuos en un área heterogénea.

Sin embargo el detalle provisto por los modelos orientados a individuos (MOI) tiene también desventajas: los modelos solo pueden ser analizados a través de simulaciones por computadora. En consecuencia es necesario contar con herramientas de hardware y herramientas de software para analizar el MOI. Otro problema técnico existente en los MOI de sistemas ecológicos es la difusión espacial, observada en enfermedades epidémicas, parásitos o virus en una estructura heterogénea de diferentes poblaciones de organismos. Por esta razón, muchos modelos [2] están basados en una lattice para la descripción de la propagación regional.

Capítulo II. Marco Teórico

2.1. Autómatas celulares

El desarrollo de los autómatas celulares (AC), tiene como punto de inicio los esfuerzos del matemático polaco Stanislaw Marcin Ulam en el laboratorio nacional de los Álamos. Ulam estudiaba el crecimiento de cristales mediante un modelo abstracto que consistía en una lattice. Al mismo tiempo John Von Neumann consideraba la posibilidad de modelar el comportamiento del cerebro mediante un autómata. Sin embargo, von Neumann argumentaba que tal autómata debería poseer además mecanismos de auto – control y auto – replicación. Por consejo de Ulam, von Neumann decidió modelar los mecanismos de auto – replicación mediante un universo discreto compuesto de "células". Fue así que los esfuerzos de von Neumann se enfocaron en hallar una abstracción de los mecanismos de auto – replicación.

En el universo de células, estas son caracterizadas por un estado interno, el cual consiste de un número finito de bits de información. Von Neumann sugirió que este sistema de células debería evolucionar en etapas discretas de tiempo, al igual que un autómata simple calcularía su nuevo estado a partir de una regla simple. De manera similar a lo que ocurre en un sistema biológico, la actividad de las células se lleva a cabo de manera simultánea.

El primer AC de von Neumann estaba compuesto de una lattice cuadrada de varios miles de células elementales. Cada una de estas células podía estar en uno de 29 estados distintos. La regla de evolución consideraba cada célula y sus 4 vecinas ortogonales, arriba, abajo, a la izquierda y a la derecha (Figura 1). Debido a la complejidad de este autómata, este solo ha sido parcialmente implementado en una computadora [3].

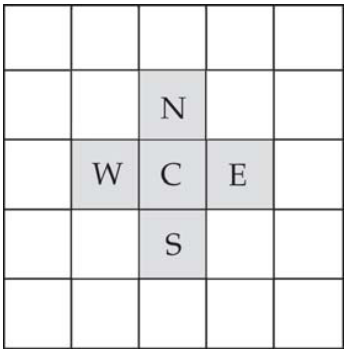


Figura 1. Vecindad de Neumann.

En la década de 1960 un AC de 8 estados fue desarrollado por el matemático británico Edgar F. Codd. En 1964 Christopher Langton demostró la $auto - replicación$ en un autómata en forma de bucle de 86 células utilizando 8 estados y una vecindad de 5 células (Figura 2). Sin embargo este autómata no exhibía la propiedad de cómputo universal [4].

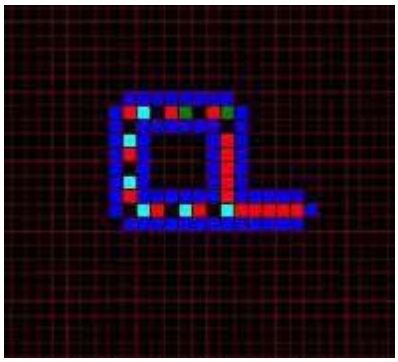


Figura 2. Autómata de Langton.

Muchos AC interesantes han surgido desde entonces; algunos como juegos de computadora, que gracias a las facilidades computacionales y a las diversas teselaciones del plano, es posible aplicar reglas locales que dan lugar a vistosos cambios en las configuraciones; tal es el caso del llamado "juego de la vida" presentado por el matemático británico John Horton Conway, en la columna "Mathematical Games" de Scientific American, en octubre de 1970. El juego de la vida es un AC bidimensional, el cual consiste de un plano dividido en pequeños cuadrados llamados células, algunos de los cuales se sombrean. Las células sombreadas son células vivas, las no sombreadas son células muertas. Cada célula tiene ocho células vecinas, que corresponden a las células adyacentes en los diferentes puntos cardinales: N, S, E, O, NE, NO, SE y SO.

El juego de la vida funciona bajo las siguientes reglas las cuales se aplican simultáneamente a todas las células:

- Si una célula muerta tiene exactamente tres células vecinas vivas, entonces nace una célula en el lugar de la célula muerta.
- Si una célula viva tiene dos o tres células vecinas vivas, la célula se conserva viva, pero si hay menos de dos células vecinas vivas, la célula muere por aislamiento.
- Si una célula tiene más de tres células vecinas vivas, muere por sobrepoblación.

Actualmente los AC se utilizan ampliamente para modelar sistemas físicos, biológicos, así como sistemas de eventos discretos. Además, se han utilizado para simular un amplio número de procesos naturales tales como: flujo turbulento, difusión de gases, incendios forestales y avalanchas.

2.1.1. Descripción de un AC

Un AC es un sistema dinámico discreto que es adecuado para modelar sistemas naturales que pueden ser descritos por una colección masiva de objetos simples que interactúan localmente entre sí [5].

Un AC consta de los siguientes elementos:

- Una retícula o lattice $d - dimensional$ infinitamente extendida que representa el espacio; donde d es un entero no negativo; usualmente para implementaciones en

computadora los valores de d son: $0, 1, 2, 3$. Cada localidad de la lattice es llamada célula. En el caso unidimensional, las células forman una fila de células adyacentes. Idealmente el número de células es infinito, pero para propósitos prácticos, se considera lo suficientemente grande para ilustrar el comportamiento del sistema a modelar.

- Cada célula de la lattice tiene un valor asociado conocido como estado, el cual puede tomar un conjunto finito de valores. Si cada célula de la lattice puede tomar su valor del mismo conjunto de estados, diremos que la lattice es homogénea.
- Una función de transición que gobierna la forma en la que cada célula altera su estado actual en el siguiente instante de tiempo. Esta función toma en cuenta el estado actual de la célula, así como el de aquellas que se encuentren en su vecindad, y regresa el nuevo estado de la célula.

2.1.2. Condiciones de frontera

Para fines prácticos es imposible diseñar un AC que posea una lattice infinita de células. En consecuencia, es necesario considerar algunas condiciones de frontera que gobiernen las interacciones de las células que se ubican en los extremos de la lattice. Las condiciones de frontera más comunes son las siguientes:

- **Frontera Periódica**. En este tipo de frontera, las células en los extremos izquierdo y derecho se consideran conectadas. En el caso unidimensional es posible visualizar la lattice como un anillo o lazo cerrado. En el caso bidimensional es común considerar, además los extremos inferior y superior conectados. En este caso la lattice toma la forma de un toroide (Figura 3).

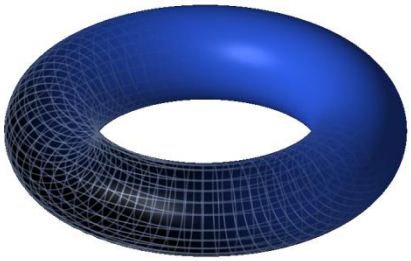


Figura 3. Toroide.

- **Frontera Fija**. En algunos casos se desea que las células en los extremos de la lattice tomen algún valor previamente establecido, ejemplo de esto es la modelación de fenómenos relativos a la conducción de calor. En consecuencia, los valores de estas células permanecen siempre fijos durante toda la simulación.
- **Frontera Reflectora**. Los valores de las células en los bordes de la lattice se consideran reflejados fuera de esta [6].
- **Sin Frontera**. La lattice comienza con una dimensión finita, y el espacio crece dinámicamente conforme se requiere [6].
- **Combinación de las anteriores**. En algunos casos podemos considerar los extremos de la Lattice con alguno de los tipos de frontera anteriormente citados, y que en otro extremo

podría tener alguna otra condición. Esto le dará al Lattice una geometría específica, por ejemplo, si para una Lattice en dos dimensiones se maneja una frontera periódica en dos de sus extremos opuestos y una frontera abierta en los otros dos extremos, el Lattice puede visualizarse en tres dimensiones como una banda circular [6].

2.1.3. Definición de AC

Un AC, es una tupla (L, S, N, f) donde:

- L es una lattice de dimensión d, con $d \in \mathbb{Z}^+$. En el caso de una lattice finita, existe un número finito de células, y es implementada utilizando alguna de las condiciones de frontera mencionadas anteriormente.
- $S = \{0,1,2, \dots, k-1\}$ es un conjunto finito de estados.
- $N = \{N(c): c \in L \text{ y } N(c) \text{ es la vecindad de c de tamaño } r \in \mathbb{Z}^+ \ \forall c \in L\}$ es un conjunto de vecindades.
- Una vecindad de tamaño $r \in \mathbb{Z}^+$ para una célula c es un conjunto finito de células $N(c) = \{c_1, \dots, c_n\} \subset L,\ c = c_j$ para algún j si $c \in N(c)$ o para ningún j en caso contrario.
- $f: S^n \to S$ es una función de transición local. Esta función se aplica en cada etapa de tiempo discreto sobre las células de L. Esta función toma como argumentos los estados de las células de $N(c)$ y regresa como resultado el nuevo valor de c en el siguiente instante de tiempo.

Se dice que un $AC = \{L, S, N, f\}$ es homogéneo, si y solamente si:

a) L es homogénea.
b) Si f se aplica en cada etapa de tiempo discreto sobre toda célula de L por igual.

Si al menos se cumple 1, se dice que A es un AC de lattice homogénea. Cualquier asignación de valores a los estados de cada célula mediante la función de transición resulta en una configuración particular en un instante de tiempo t (Figura 4).

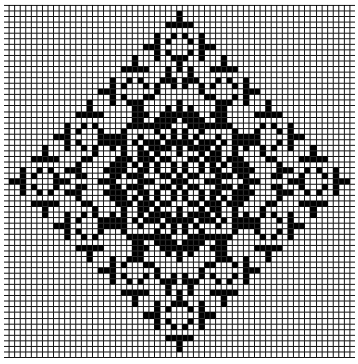


Figura 4. Una configuración bidimensional[1] [4].

[1] La primera referencia impresa de un sistema como este, con el uso de la palabra autómata se encuentra en el artículo de Ulam de 1950.

Dada una configuración, la asignación de nuevos estados a todas las células para generar una nueva configuración, a través de la función de transición de A es conocida como función global.

Sea $A = (L, S, N, f)$ un AC. La función global es una función $G: S^L \rightarrow S^L$ que se aplica evaluando f sobre cada una de las células de A en el tiempo t (es decir, sobre una configuración C_t de A en el tiempo t) y regresa nuevos valores para todas las células de A (es decir, se pasa de la configuración C_t de A a la configuración C_{t+1} de A, desde t a $t + 1$.

2.1.4. Evolución de un AC unidimensional.

En el caso del análisis de un AC unidimensional, es común definir la vecindad de una célula c como un conjunto de tres células compuesto por la misma c, y aquellas células que se encuentran a su izquierda y derecha. Además $S = \{0,1\}$, es decir, el conjunto de estados posibles para cada célula es binario. En consecuencia, una vecindad de tres células con estados permisibles 0 y 1 para cada una, puede ser expresada de $2^3 = 8$ diferentes formas (Figura 5).

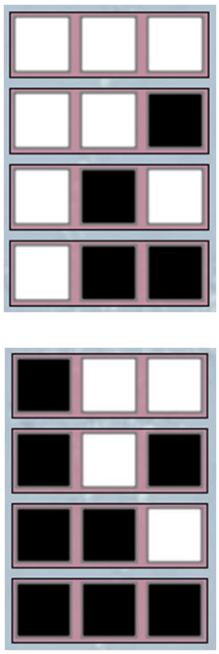


Figura 5. Los ocho posibles estados para la vecindad de una célula c.

Por convención $c_i(t)$ denota el estado de la $i - ésima$ célula en el tiempo t. En la siguiente etapa de tiempo $t + 1$, el estado de la célula será $c_i(t + 1)$. Matemáticamente podemos expresar la dependencia del estado de una célula en la etapa de tiempo $t + 1$, de los estados de las células $c_{i-1}(t)$ y $c_{i+1}(t)$ como:

$$c_i(t + 1) = f(c_{i-1}(t), c_i(t), c_{i+1}(t))$$

Donde f es la función de transición local. Una manera conveniente de ilustrar las reglas permisibles para un AC unidimensional es indicar el estado (color) de la célula de en medio en la

siguiente etapa de tiempo, dado el estado (color) de sí misma y el de sus vecinas (Figura 6). La célula de en medio (fila inferior) representa el nuevo estado de la célula en $t + 1$.

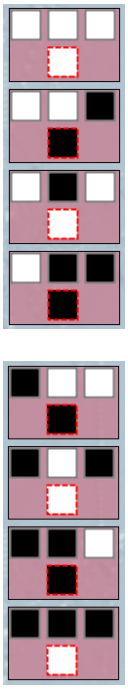


Figura 6. Representación gráfica de las reglas permisibles de un AC unidimensional.

La Figura 7 es la representación gráfica del AC gobernado por la regla anterior (Figura 6). Podemos observar la célula inicial en el centro y el patrón resultante con la generación de cada nueva fila.

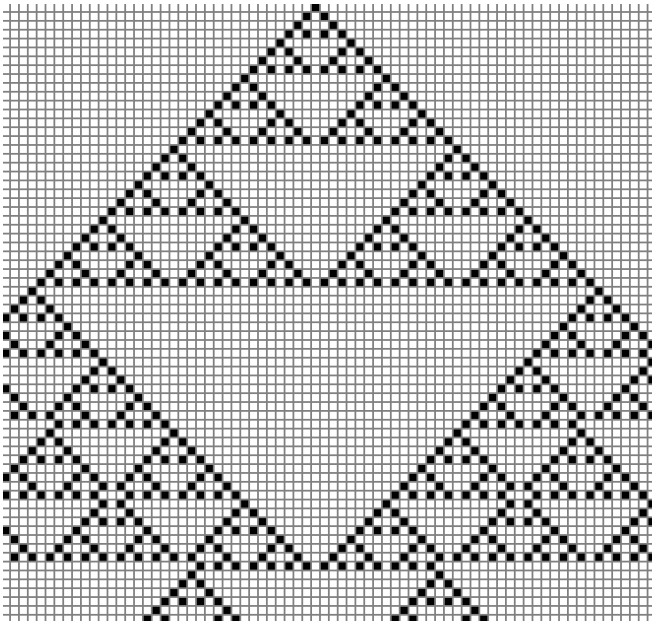


Figura 7. Triángulo de Sierpinski, generado a partir de un AC unidimensional.

Dado que existen 8 diferentes estados para la vecindad de este autómata, y como cada célula tiene 2 estados posibles, existen $2^8 = 256$ posibles reglas de transición. Wolfram denominó a cada uno de los AC resultantes de estas reglas como $Automatas\ Celulares\ Elementales$. Si consideramos el estado de una célula blanco como 0 y el estado de una célula negra como 1; podemos asignar el número $01011010_{bin} = 90_{dec}$ a la regla de la Figura 6.

2.2.Redes de Petri

Las Redes de Petri tienen como origen la tesis doctoral del matemático alemán Carl Adam Petri (Figura 8) "Kommunikation mit Automaten" (Comunicación con Autómatas) de 1962. El trabajo de Petri no se centró en resolver un problema abierto o elaborar una nueva teoría, sino en presentar varias ideas y propuestas con el fin de revisar los fundamentos de la informática. Petri comienza con el análisis de un problema conocido en la década de 1960 y es el siguiente: dada una función recursiva f y un argumento n, ¿Es posible estimar la cantidad de recursos necesarios para computar $f(n)$? Típicamente, el computo inicia con una cantidad finita de recursos de computo, si estos recursos son suficientes, el computo de $f(n)$ termina en algún punto. De lo contrario, es necesario reunir más recursos y reiniciar el computo de $f(n)$. En su tesis Petri plantea la posibilidad de organizar un sistema de cómputo en el cual sea posible colocar nuevos componentes, cada vez que el computo de $f(n)$ lo requiera, sin que esto impacte el tiempo de computo de manera significativa. Petri no solo demostró que en arquitecturas convencionales el proceso anterior es imposible; además, mostró el modelo de lo que sería una arquitectura extensible [7]. En esta arquitectura cada componente debe ser capaz de actuar de manera autónoma y en consecuencia el sistema entero debe trabajar asíncronamente.

Figura 8. Carl Adam Petri.

Las ideas de Petri llamarón la atención de un grupo de investigadores dirigidos por Anatol Holt. Este grupo buscaba un modelo para representar y analizar sistemas, así como su comportamiento. Es el trabajo de este grupo de investigadores el que sienta las bases de los fundamentos teóricos, así como la notación y la representación gráfica de las Redes de Petri [8]. Además dado el trabajo previo de Petri en el modelado de arquitecturas asíncronas, este grupo mostró como las Redes de Petri podían ser aplicadas al análisis de sistemas concurrentes.

A partir del trabajo inicial de Petri ha existido gran interés en el desarrollo teórico y potenciales aplicaciones de las Redes de Petri [9].

Debido a la eficiencia de las Redes de Petri para describir sistemas, estas han sido aplicadas con éxito en áreas tales como: los sistemas de potencia [10], el diseño de modelos para el análisis de controladores de sistemas de eventos discretos [11], así como el análisis estructural de redes biológicas [12].

2.2.1. Descripción de una Red de Petri

Las Redes de Petri (RP) son una herramienta matemática cuyo propósito es describir, modelar y analizar sistemas que pueden ser caracterizados como concurrentes, asíncronos, distribuidos, paralelos, no deterministas y/o estocásticos. Debido a la naturaleza matemática de las RP, un sistema puede ser descrito por un conjunto de ecuaciones algebraicas lineales. Este proceso facilita el análisis formal de cualquier sistema, ya que las propiedades que definen el comportamiento del sistema bajo análisis pueden ser representadas por alguna de estas ecuaciones.

En general una RP puede ser descrita como un grafo bipartito dirigido formado por 4 tipos de objetos:

- Sitios. Representados gráficamente por un circulo (Figura 9). Un sitio es una entrada a una transición, si existe un arco dirigido conectando el sitio a la transición. De manera similar, un sitio es una salida de una transición, si existe un arco dirigido que conecta la transición al sitio. Dependiendo del sistema a ser modelado un sitio puede ser interpretado como una condición inicial, datos de entrada, recursos necesarios, etc.

Figura 9. Representación gráfica de un sitio.

- Transiciones. Representados gráficamente por barras o cajas (Figura 10). Dependiendo del sistema bajo análisis, una transición puede ser interpretada como un evento, una etapa en un algoritmo, una señal en un microprocesador, una tarea, una clausula, etc.

Figura 10. Representación gráfica de una transición.

- Arcos. Representados gráficamente por una flecha (Figura 11). Describen la relación entre sitios y transiciones [13]. Generalmente sirven para identificar entradas y salidas de una transición.

Figura 11. Representación gráfica de un arco.

- Marcas (Tokens). Representadas gráficamente por puntos dentro de un sitio (Figura 12). Para estudiar el comportamiento dinámico del sistema modelado en términos de sus estados y sus cambios, cada sitio puede mantener ya sea cero o un número positivo de marcas. La presencia o ausencia de una marca en un sitio puede ser interpretada como una condición asociada a ese sitio, por ejemplo, el ser verdadera o falsa.

Figura 12. Representación gráfica de una marca (token).

La Figura 13 muestra la representación gráfica de una RP. Esta red consiste de 5 sitios, 4 transiciones y arcos dirigidos que conectan sitios a transiciones y transiciones a sitios. En esta red, p_1 es un sitio de entrada de la transición t_1. Los sitios p_2 y p_3 son sitios de salida de la transición t_1.

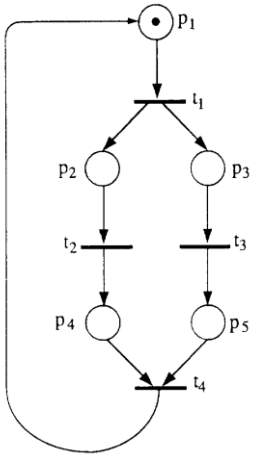


Figura 13. Representación gráfica de una RP [14].

2.2.2. Definición de una RP

Una RP es una tupla (P, T, F, W, M_0) donde:

- $P = \{p_1, p_2, \dots, p_m\}$ es un conjunto finito de sitios.
- $T = \{t_1, t_2, \dots, t_n\}$ es un conjunto finito de transiciones.
- $F \subseteq (P \times T) \cup (T \times P)$ es un conjunto de arcos (relación de flujo).

- $W: F \rightarrow \{1,2,3, \dots\}$ es una función de peso.
- $M_0: P \rightarrow \mathbb{N} \cup \{0\}$, es el marcado inicial.
- $P \cap T = \emptyset$ y $P \cup T \neq \emptyset$.

Para poder analizar el comportamiento de un sistema, el marcado de una RP cambia de acuerdo a la siguiente regla de transición ($firing\ rule$):

a) Una transición t se considera habilitada si cada sitio de entrada p de t está marcado con al menos $w(p,t)$ marcas, donde $w(p,t)$ es el peso del arco de p hacia t.
b) Una transición habilitada puede o no disparar, dependiendo si el evento representado por la transición toma lugar o no.
c) El disparo de una transición habilitada t remueve $w(p,t)$ marcas de cada sitio de entrada p de t, y agrega $w(t,p)$ marcas a cada sitio de salida p de t, donde $w(t,p)$ es el peso del arco de t a p.

Una transición que no tiene sitios de entrada es llamada transición fuente. Mientras que una transición que no tiene sitios de salida es llamada transición sumidero (sink transition). Una transición fuente está incondicionalmente habilitada. El disparo de una transición sumidero consume las marcas de sus sitios de entrada pero no produce ninguna.

El par formado por un sitio p y una transición t es llamado auto-lazo ($self-loop$), si p es tanto un sitio de salida como un sitio de entrada a t. Una RP es llamada pura si la misma no contiene auto-lazos. Una RP es ordinaria si todos los arcos que la componen tienen un peso igual a uno [9].

La Figura 14 ilustra el comportamiento de la regla de disparo. En la Figura 14a, la transición t_1 está habilitada ya que su sitio de entrada p_1 contiene dos marcas y $w(p_1,t_1) = 2$. El disparo de t_1 remueve de p_1 dos marcas, ya que $w(p_1,t_1) = 2$. En la Figura 14b se muestra la consecuencia del disparo de t_1, una marca es depositada en p_3, ya que $w(t_1,p_3) = 1$ y dos marcas son depositadas en p_2, ya que $w(t_1,p_2) = 2$.

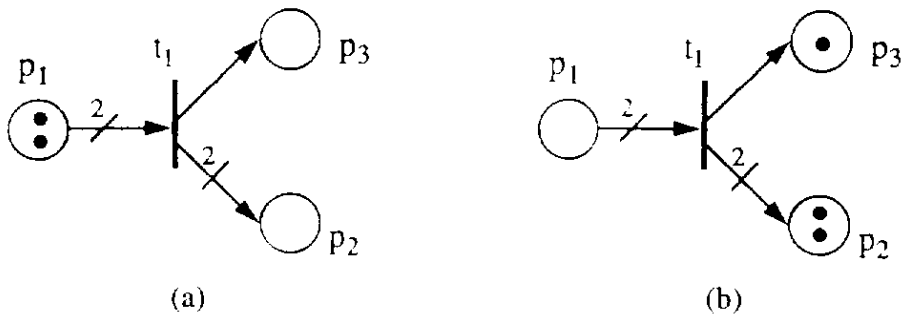


Figura 14. Habilitación y disparo de la transición t_1.

Ejemplo (adaptado de [15]): Considere un sistema de cómputo para procesamiento en serie. Supongamos que este tiene 4 estados o condiciones:

a) Una tarea espera a ser procesada.
b) El procesador se encuentra inactivo.
c) Una tarea está siendo procesada.
d) Una tarea espera acceso a la salida del sistema.

El sistema puede tomar 4 posibles acciones:

a) Una tarea entra a la cola de entrada.
b) Empieza la ejecución de una tarea en el procesador.
c) Una tarea es completada.
d) Una tarea es puesta en la salida del sistema.

Si una tarea entra a la cola de entrada, entonces la misma se considera como una tarea en espera; si una tarea se encuentra en espera y el procesador está inactivo, entonces puede iniciar la ejecución de una tarea en el procesador. Una vez que una tarea ha sido procesada, esta es completada y la misma espera acceso a la salida del sistema; además el procesador entra en un estado de inactividad. Si una tarea está esperando acceso a la salida del sistema, se le otorga.

El sistema anterior puede ser descrito mediante una RP con cuatro sitios: p_1, p_2, p_3, p_4, que corresponden a los cuatro estados, y cuatro transiciones t_1, t_2, t_3, t_4, que corresponden a las cuatro acciones que puede desempeñar el sistema. La descripción formal del sistema es la siguiente:

$RP = (P, T, F, W, M_0)$, donde:

- $P = \{p_1, p_2, p_3, p_4\}$
- $T = \{t_1, t_2, t_3, t_4\}$
- $F = \{(p_1, t_2), (p_2, t_2), (p_3, t_3), (p_4, t_4), (t_1, p_1), (t_2, p_3), (t_3, p_2), (t_3, p_4)\}$
- $W: F \to \mathbb{N} \cup \{0\}$
- $M_0 = \{0,1,0,0\}$

La función de peso W se define de acuerdo a lo siguiente:

- $W(p_1, t_2) = 1$
- $W(p_2, t_2) = 1$
- $W(p_3, t_3) = 1$
- $W(p_4, t_4) = 1$
- $W(t_1, p_1) = 1$
- $W(t_2, p_3) = 1$
- $W(t_3, p_2) = 1$
- $W(t_1, p_1) = 1$

El diagrama para la RP anterior es el mostrado en la Figura 15.

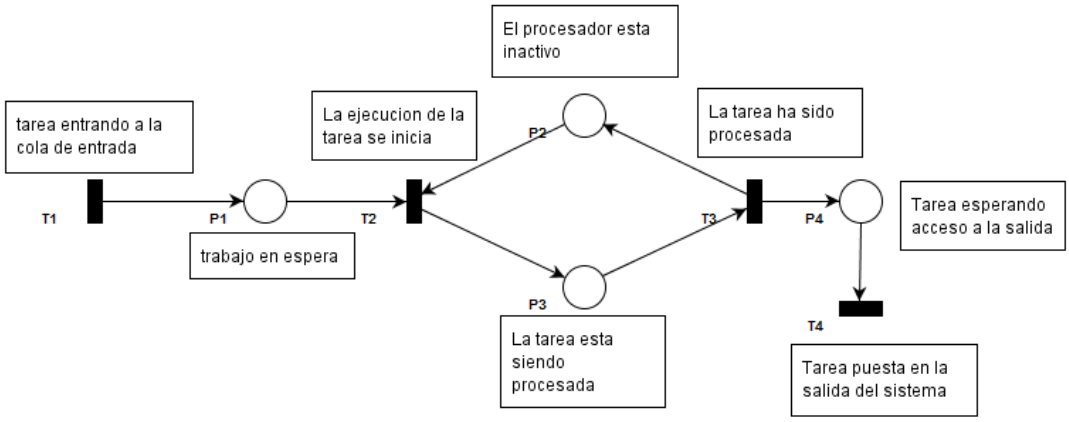


Figura 15. Diagrama de la RP para el ejemplo propuesto.

2.2.3. Redes de Petri Estocásticas

Típicamente las Redes de Petri no incluyen un análisis temporal, ya que no se define el instante de tiempo en que una transición puede disparar, en consecuencia, no es posible analizar el desempeño de una RP. Con la definición actual solo es posible realizar el análisis cualitativo de un sistema. Para analizar el desempeño de un sistema, es necesario agregar una componente temporal a la definición de una RP.

Existen dos métodos principales para incorporar tiempo en una RP:

a) Especificando un tiempo de residencia en un sitio para las marcas. Si las marcas son disparadas hacia un lugar p, estas permanecerán no disponibles para todas las transiciones de salida de p por un intervalo de tiempo. Una vez que este tiempo ha expirado, las marcas estarán disponibles y podrán ser consumidas por las transiciones de salida de p. Estas RP son conocidas como RP de Sitios Temporizados ($Timed\ Places\ Petri\ Nets - TPPNs$) [16].

b) Especificando un intervalo de disparo para las transiciones habilitadas. Una vez que una transición ha sido habilitada, esta disparará después de que un cierto intervalo de tiempo haya pasado. Estas RP son conocidas como RP de Transiciones Temporizadas ($Timed\ Transitions\ Petri\ Nets - TTPNs$) [16]. Este tipo de redes pueden ser clasificadas a su vez en dos grupos:
   - Modelos de Preselección. Una vez que una transición ha sido habilitada, esta selecciona todas las marcas de sus sitios de entrada que necesita para disparar. Las marcas seleccionadas son inaccesibles para cualquier otra transición de la RP. La transición habilitada espera hasta que su intervalo de disparo haya expirado, momento en el cual dispara inmediatamente. Al disparar, la transición destruye las marcas seleccionadas y crea marcas en sus sitios de salida de acuerdo a la regla de disparo de la RP.

- Modelos de Competencia. En estos modelos las marcas no son reservadas por una transición. Una vez que una transición ha sido habilitada, esta espera a que su intervalo de disparo expire, si en este instante la transición aún está habilitada disparará inmediatamente. Dado que una transición más rápida puede deshabilitar a otras transiciones, es claro que todas las transiciones de la red "compiten" por las marcas. De aquí el nombre de este modelo.

2.2.4. Redes de Petri y autómatas

Un lenguaje puede ser considerado como una manera formal de describir el comportamiento de un sistema de eventos discretos (DES). Este especifica todas las secuencias de eventos admisibles que el sistema es capaz de procesar.

Sea E el conjunto de eventos asociados a un DES, este conjunto puede ser visto como el "alfabeto" del lenguaje asociado al sistema. Las secuencias de eventos admisibles por el mismo constituyen palabras del lenguaje. El conjunto de todas la cadenas de longitud finita formadas a partir de la ocurrencia de los eventos en E constituyen el lenguaje L. Un autómata es un dispositivo capaz de representar un lenguaje de acuerdo a reglas bien definidas.

Un autómata finito determinista G está definido por la tupla: $(X, E, f, \Gamma, x_0, X_m)$, donde:

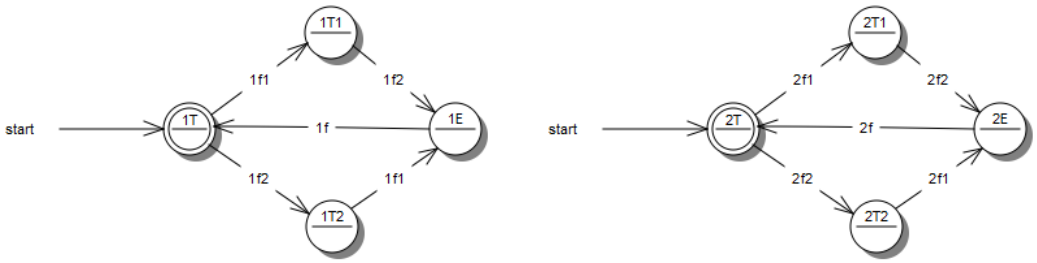
- X es un conjunto finito de estados.
- E es el conjunto finito de eventos asociados con G.
- $f: X \times E \to X$ es la función de transición, $f(x, e) = y, x \in X \wedge e \in E$, significa que existe una transición etiquetada por el evento e, desde el estado x al estado y.
- $\Gamma: X \to 2^E$, $\Gamma(x)$ es el conjunto de todos los eventos e para los cuales $f(x, e)$ está definida. Este conjunto es llamado conjunto de eventos activos (o conjunto de eventos factibles) de G en x.
- x_0 es el estado inicial.
- $X_m \subseteq X$ es el conjunto de estados marcados. Los estados son marcados cuando se desea asociar un significado especial a ellos. Es común referirse a estos estados como estados de "aceptación" o estados "finales".

El autómata G opera de la manera siguiente: comienza en un estado inicial x_0, y tras la ocurrencia de un evento $e \in \Gamma(x_0) \subseteq E$ efectuará una transición al estado $f(x_0, e) \in X$. Este proceso continua en base a las transiciones para las cuales f está definida.

Para ejemplificar el uso de un autómata finito considere una situación donde varios usuarios comparten un recurso en común. Una forma de ejemplificar esto es a través del conocido problema de los "filósofos cenando", para mantener la claridad, reduciremos el problema a solo dos filósofos. Estos filósofos se encuentran cenando en una mesa redonda, donde existe un plato de comida para cada filósofo, y dos tenedores uno a cada lado entre los platos.

El comportamiento de los filósofos es como sigue: cada filósofo puede estar pensando o comiendo. Para pasar del estado "pensando" al estado "comiendo", un filósofo debe tomar ambos tenedores, uno a la vez, en cualquier orden. Una vez que el filósofo ha terminado de comer, deja ambos tenedores sobre la mesa, y regresa al estado "pensando". La Figura 16 muestra los autómatas finitos que modelan el comportamiento de los filósofos. La notación de los diagramas sigue la siguiente convención:

- El evento $if j$ indica que el filósofo i toma el tenedor j.
- El evento if indica que el filósofo i deja ambos tenedores sobre la mesa.
- El estado iT indica que el filósofo i está en el estado "pensando".
- El estado iE indica que el filósofo i está "comiendo".
- El estado iTj indica que el filósofo i está en el estado "pensando" y tiene posesión del tenedor j.



(a) P1 (b) P2

Figura 16. Autómatas finitos modelando el comportamiento de un filósofo.

Para completar el comportamiento del problema es necesario construir dos autómatas que modelen la transición de estados de los tenedores (Figura 17). En ambos diagramas la notación iA indica que el tenedor i está disponible, de manera similar iU indica que el tenedor i está en uso.

(a) F1 (b) F2

Figura 17. Comportamiento de los tenedores modelado por autómatas finitos.

Para generar el autómata que modela en su totalidad el problema de los filósofos cenando, es necesario recurrir a la composición paralela de los autómatas que modelan a los filósofos y a los tenedores. Cada uno de estos autómatas puede o no tener eventos privados, i. e., eventos que modelan su comportamiento interno, además existirán eventos comunes, estos eventos son compartidos por los autómatas y capturan el acoplamiento entre los respectivos componentes del sistema.

La composición paralela de dos autómatas finitos G_1 y G_2 se define como:

$$G_1 \| G_2 = Ac(X_1 \times X_2, E_1 \cup E_2, f, \Gamma_{1\|2}, (x_{01}, x_{02}), X_{m_1} \times X_{m_2}) \qquad (2.1)$$

En la composición paralela, un evento común, esto es, un evento en $E_1 \cap E_2$, solo puede ser ejecutado si los autómatas G_1 y G_2 lo hacen de manera simultánea. Los eventos privados, aquellos en $(E_2 \setminus E_1) \cup (E_1 \setminus E_2)$, no están sujetos a esta restricción y pueden ser ejecutados siempre que sea posible [17].

En la Figura 18 se muestra la composición paralela $P_1 \| P_2 \| F_1 \| F_2$ que modela el caso específico de dos filósofos cenando.

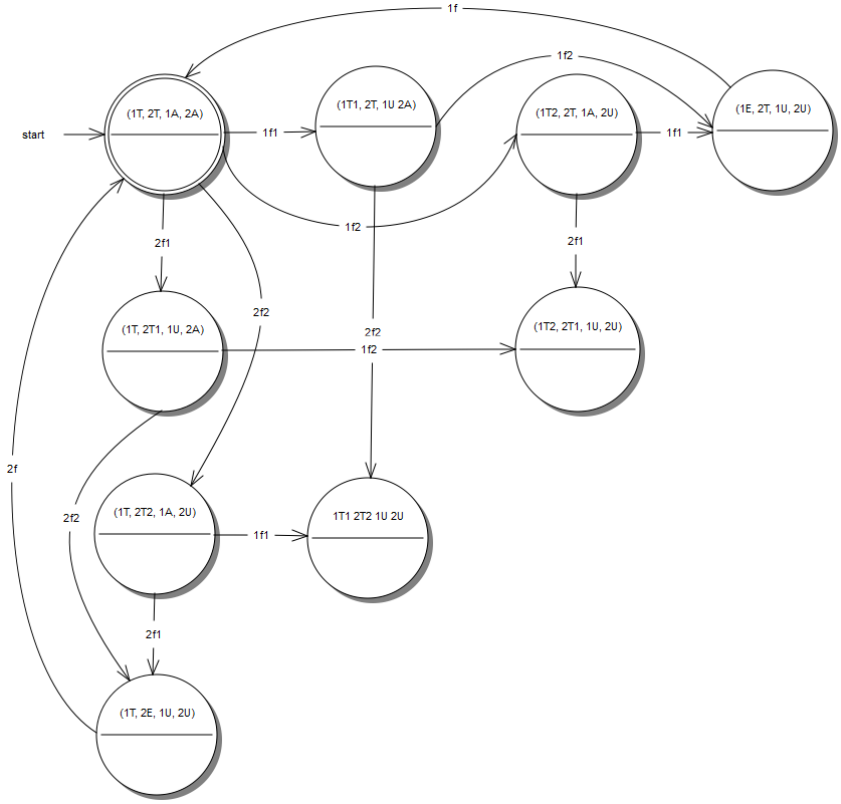


Figura 18. Composición paralela para el problema de dos filósofos cenando.

Al igual que los autómatas, las redes de Petri pueden ser utilizadas para representar el comportamiento de un DES. En un autómata esto se hace enumerando explícitamente todos los estados posibles, y conectando estos estados con las transiciones entre ellos, dando lugar así a la función de transición del autómata. Una ventaja de esta aproximación es que se pueden utilizar operaciones como la composición paralela para crear modelos de sistemas complejos, a partir de modelos más simples de manera sistemática.

En contraste, en las redes de Petri los estados no son enumerados, esta información es distribuida en los sitios de la red, de esta manera se capturan las condiciones clave que gobiernan la operación de un sistema. Una ventaja inherente de las redes de Petri es su habilidad para descomponer o sintetizar un sistema complejo. Suponga que se tienen dos sistemas con sus respectivos conjuntos de estados X_1 y X_2 modelados a través de un autómata. Si combinamos estos dos sistemas en uno, su conjunto de estados puede llegar a ser tan grande como $X_1 \times X_2$

sobre todo, si los dos sistemas no tienen eventos en común. Por otro lado, si los sistemas fueran modelados a través de una red de Petri, el proceso de combinarlos generalmente solo involucra agregar la interface necesaria (sitios y transiciones) que represente el acoplamiento de ambos sistemas.

Considere nuevamente el ejemplo de los dos filósofos cenando, la red de Petri que modela el comportamiento de un filósofo se muestra en la Figura 19, la notación utilizada en esta red es análoga a la utilizada para el ejemplo de los autómatas. En esta red existe un sitio para el estado "comiendo" (iE), así como un sitio para el estado "pensando" (iT). La red modela la disponibilidad de los tenedores a través de los sitios iA, la presencia de una marca en estos sitios indica que el tenedor i está disponible.

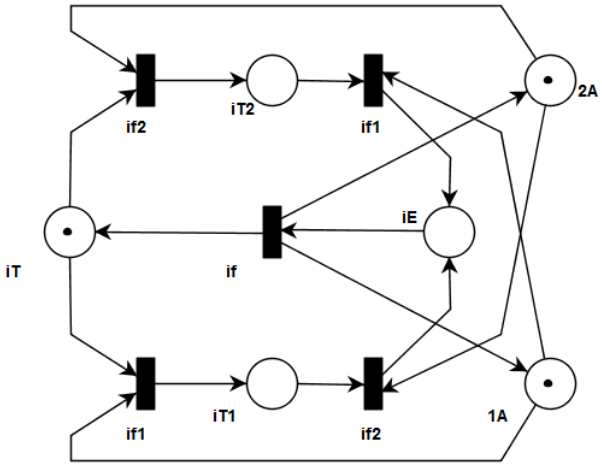


Figura 19. Red de Petri para un filósofo.

Para obtener la red de Petri que modela el comportamiento de ambos filósofos, basta con crear una copia de la red de la Figura 19 y combinar los sitios iA de ambas redes (Figura 20). La representación de este problema a través de una red de Petri permite una construcción modular y una descripción intuitiva del problema a través del análisis de la misma.

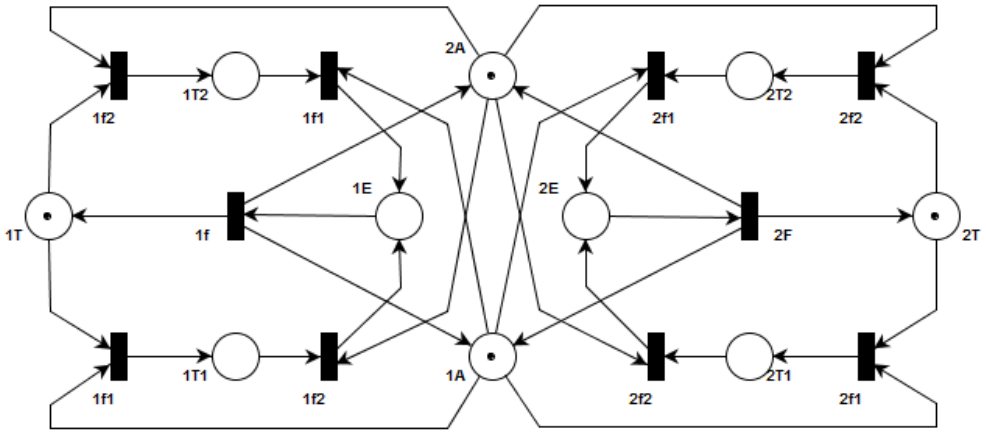


Figura 20. Red de Petri para el problema de dos filósofos cenando.

A continuación se listan las diferencias más relevantes entre los autómatas finitos y las redes de Petri:

- Los autómatas finitos representan una manera rápida de modelar sistemas simples.
- Las redes de Petri facilitan la construcción y análisis modular de un sistema complejo.
- En un autómata finito, el número de estados necesario para modelar un sistema crece rápidamente, sobre todo al combinar sistemas que no tienen eventos en común.
- A través de una red de Petri, el comportamiento de un sistema es descrito de manera intuitiva.
- En un autómata finito es necesario enumerar de manera explícita los estados por los que puede atravesar el sistema.
- En una red de Petri, la información acerca del estado de una red es distribuida en todos los sitios de la misma (el estado de una red está definido por su marcado en un momento dado).

2.3.Autómatas probabilísticos

2.3.1. Distribuciones de probabilidad

Sea Ω un conjunto. Una función $\mu: \Omega \to [0,1]$ es llamada distribución de probabilidad discreta sobre Ω si $\{x \in \Omega : \mu(x) > 0\}$ es finito, o infinito contable, y $\sum_{x\in\Omega} \mu(x) = 1$, el conjunto $\{x \in \Omega : \mu(x) > 0\}$ es llamado soporte de μ y se denota por $spt(\mu)$. Si $x \in \Omega$, entonces μ_x^1 denota la distribución de probabilidad única $\mu_x^1(x) = 1$. El conjunto de todas las distribuciones de probabilidad discreta sobre el conjunto Ω es denotado por $D(\Omega)$. Si μ es una distribución con soporte finito $\{s_1, \dots, s_n\}$, es común escribir $\{s_1 \mapsto \mu(s_1), \dots, s_n \mapsto \mu(s_n)\}$ para denotar las probabilidades individuales de los elementos del $spt(\mu)$.

2.3.2. Sistemas de transición etiquetados

Los sistemas de transición de estados (Transition Systems – TS) representan la forma más simple de modelar el comportamiento de un sistema dinámico, o de un sistema de eventos discretos. La definición de un TS incluye un conjunto de estados S y una función de transición $\alpha: S \to P(S)$.

Como se mostró en la sección 2.2.4, un cambio de estado es comúnmente consecuencia de un evento asociado al sistema modelado. En este sentido, los sistemas de transición etiquetados (Labeled Transition Systems – LTS) representan una evolución natural de los TS, esto se hace comúnmente al etiquetar las transiciones entre estados con símbolos que representan la ocurrencia de algún evento. Entonces, es posible definir un LTS mediante la tupla (S, E, α), donde:

- S es un conjunto de estados.
- E es un conjunto de eventos.

- $\alpha: S \to P(E \times S)$

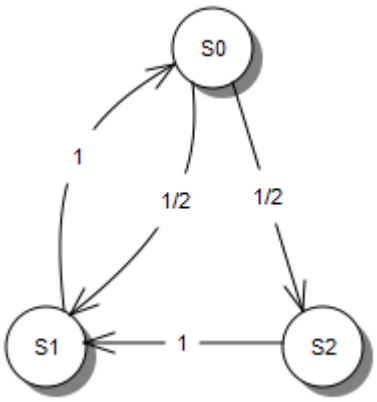
Para cada estado $s \in S$ de un LTS, cada elemento $(e, s') \in \alpha(s)$ determina una transición que es denotada por $s \xrightarrow{e} s'$.

2.3.3. Cadenas de Markov discretas en tiempo

Las cadenas de Markov discretas en tiempo representan la clase más simple de autómatas probabilísticos [18]. Una cadena de Markov discreta en tiempo es una tupla (S, α), donde:

- S es un conjunto de estados.
- $\alpha: S \to D(S)$ es una función de transición.

Las cadenas de Markov se derivan de manera intuitiva a partir de modelos más simples de transición de estados, cuando una probabilidad se agrega a cada transición de manera tal que, para cada estado, la suma de las probabilidades de las transiciones salientes es igual a 1 (Figura 21). La clase de todas las cadenas de Markov es denotada por MC. Si $s \in S$, $\alpha(s) = \mu$ y $\mu(s') = p > 0$, entonces se dice que la cadena de Markov (S, α) pasa del estado s con probabilidad p al estado s'.



$$S = \{s_0, s_1, s_2\}$$

$$\alpha(s_0) = \{s_0 \mapsto 0, s_1 \mapsto \frac{1}{2}, s_2 \mapsto \frac{1}{2}\}$$

$$\alpha(s_1) = \mu^1_{s_0}$$

$$\alpha(s_2) = \mu^1_{s_1}$$

Figura 21. Una cadena de Markov discreta en tiempo (Adaptado de [18]).

2.4. Estado del arte

2.4.1. Modelación ecológica

A diferencia de otras áreas de la ciencia como la física y la química, donde las variables bajo estudio están bien definidas; los sistemas ecológicos consisten de un gran número de elementos que interactúan entre sí. De estos elementos solo unos cuantos son conocidos y comprendidos de manera adecuada. Esta problemática dificulta el estudio de los sistemas ecológicos, ya que no toda la información presente en el sistema puede ser utilizada sin la necesidad de construir un modelo, que a veces resulta tan complejo como el sistema en estudio.

La construcción de modelos ecológicos se ha servido de los avances en otras áreas de la ciencia. Existen modelos ecológicos que conceptualizan especies como constituyentes químicos que interactúan de acuerdo a sistemas de ecuaciones diferenciales [19] [20]. En [21] Stephen

Wolfram argumenta que esta aproximación a la construcción de modelos se ha vuelto obsoleta, gracias al creciente poder de las computadoras modernas para determinar las secuencias de interacciones locales entre colecciones masivas de individuos, en lo que se conoce como autómatas celulares.

Los modelos ecológicos basados en CAs pueden ser clasificados en dos categorías:

- Modelos deterministas. El objetivo de estos modelos es explorar el universo completo de resultados dinámicos.
- Modelos basados en observaciones empíricas. Su objetivo es crear modelos de predicción para un ecosistema específico.

Los modelos deterministas representan la forma más básica de los modelos de CAs, este tipo de modelos fueron ampliamente discutidos por Wolfram. La dinámica de estos modelos está basada en la aplicación de reglas simples de manera síncrona a todas las células de la lattice. Este tipo de sistemas se ha utilizado de manera moderada en la modelación ecológica, debido sobre todo, a la dificultad de describir fenómenos que gobiernan la dinámica de un ecosistema a partir de reglas deterministas. Sin embargo, existen excepciones, un ejemplo aparece en [22], donde se utiliza un CA de reglas binarias totalísticas para describir la dinámica de una población y su dependencia a la densidad de individuos de una especie.

Las reglas que gobiernan a los modelos empíricos son de naturaleza probabilística, el aspecto cuantitativo de las mismas se deriva a partir de rigurosas observaciones de campo. El objetivo de estos modelos es predecir, a partir de mediciones tomadas en pequeña escala (una muestra significativa), el estado de alguna de las variantes del ecosistema en algún punto en el futuro. Un ejemplo de este tipo de modelos aparece en [23], donde Wotton utilizó un modelo de AC para comprender la dinámica de la población de un arrecife en Washington. La comunidad estaba compuesta por mejillones, algas y diatomeas (algas unicelulares). Al observar esta comunidad por más de 6 años, Wotton estimó las probabilidades para las reglas de transición entre diferentes estados y el número de estados aproximado que el modelo debía tener. Tras el estudio de campo, eligió 15 estados para representar las características de la zona litoral (ocupación, identidad de la especie, clasificación por tamaño, etc.) e incorporó los efectos de las diversas perturbaciones, temporadas y densidades de población en las probabilidades de las reglas de transición. El modelo fue capaz de recrear la dinámica de los grupos de mejillones en la comunidad de la zona litoral. Además, al eliminar de manera selectiva ciertas características del modelo, Wotton determinó que es necesario considerar de manera explícita las perturbaciones a gran escala y las interacciones locales para generar patrones espaciales similares a los observados en el ecosistema en estudio.

2.4.2. Competencia y depredación

Existen varias aproximaciones para modelar los mecanismos de competencia entre especies, sin embargo, las siguientes categorías resaltan en la teoría:

- Competencia por interferencia. En esta categoría se modelan las interacciones negativas que surgen de manera directa entre especies competidoras.
- Competencia por explotación. En esta categoría las especies compiten por la utilización de un recurso limitado.
- Competencia aparente. En esta categoría se estudia una situación en la que dos especies competidoras son afectadas por un depredador común.

En los modelos anteriores se modelan relaciones causales directas, además, los métodos utilizados para modelar se basan en condiciones de equilibrio estáticas. Los modelos basados en la teoría ecológica clásica están formulados sólo en términos de los cambios de la biomasa del ecosistema [24]. Analizar sistemas bióticos como sistemas multiniveles es necesario si las condiciones que los gobiernan han de ser descritas. Los procesos de nivel más alto determinan la estructura de los niveles más bajos, a su vez los niveles más bajos determinan el comportamiento de los niveles más altos [25]. Entre otras aplicaciones, los modelos por competencia han sido utilizados con gran éxito para analizar ecosistemas como el ejemplo descrito a continuación, o en medicina para modelar el comportamiento de virus y bacterias [26].

En [27] se describe un ecosistema donde las interacciones entre especies son dinámicas y no puede preverse una ventaja concreta para alguna especie a corto o largo plazo. Este modelo es considerado un caso especial de competencia aparente. Sin embargo, a diferencia de los modelos típicos, la disponibilidad de ambas especies para con el depredador no es síncrona; una de las especies aparece antes que la otra, aunque su disponibilidad puede traslaparse. Este modelo llamado "Depredación Secuencial" es común en la naturaleza e. g. sistemas de plantas – herbívoros, sistemas de polinización entre insectos y flores, etc.

El propósito principal de los modelos de depredación secuencial es mostrar que la disponibilidad sucesiva de especies presa puede ser un factor importante que influencia, no sólo al depredador, si no, a través de mecanismos de retroalimentación también a las especies presa. Además las interacciones locales entre individuos en el modelo resultan en interesantes estructuras espaciales que afectan la escala de tiempo ecológica y evolutiva.

2.4.3. Ecosistema de depredación secuencial

En [28] Futuyma y Wasserman presentan el estudio de campo de un ecosistema, cuyas especies de árboles dominantes son el roble escarlata (Quercus coccínea) y el roble blanco (Quercus alba). Ambas especies de roble tienen un depredador común: la larva Alsophila Pometaria (Figura 22). En este ecosistema, las larvas son dependientes de ambas especies de roble en diferentes momentos de la temporada. Esto debido principalmente a la diferencia en la disponibilidad de hojas comestibles. El roble escarlata florece 10 días antes que el roble blanco, la eclosión de los huevos de las larvas coincide aproximadamente con el florecimiento del roble escarlata. Dado que las larvas no pueden sobrevivir más de 3 días sin comida, las mismas dependen completamente de las hojas del roble escarlata en sus primeras etapas de desarrollo. Sin embargo, las larvas son incapaces de completar su desarrollo comiendo únicamente follaje maduro. En consecuencia para finalizar su desarrollo las larvas migran hacia el roble blanco.

Figura 22. Especies involucradas en el estudio de campo de Futuyma y Wasserman

El estudio revela una relación asimétrica entre larvas y robles. Tras eclosionar las larvas deben encontrar inmediatamente un roble escarlata para alimentarse. Sin embargo, la transición de las larvas al roble blanco es gradual y puede abarcar el periodo entre el florecimiento del roble blanco y el momento en que las hojas del roble escarlata se vuelven no comestibles. Ya que solo existe una generación de larvas cada temporada, el número de larvas que se alimentan de robles escarlatas es necesariamente menor al número que migra hacia robles blancos. Por otro lado, los robles blancos sufren extensamente la presencia de las larvas, esto como consecuencia del incremente en el consumo per cápita al incrementarse el peso de las larvas. Una de las características que hacen de este ecosistema un candidato excelente para su modelación es que la mayoría de los procesos que ocurren en el mismo son locales a un árbol. Por lo tanto, los AC son herramientas excelentes para este fin.

En los capítulos posteriores, haremos uso del estudio de campo de Futuyma y Wasserman para diseñar un modelo basado en ACs, que sea capaz de reproducir la dinámica poblacional de las especies presentes en este ecosistema. Además se hará uso de las redes de Petri como una herramienta formal que describa las interacciones locales que se dan en el modelo de ACs.

Capítulo III. Desarrollo y solución del problema.

3.1.Descripción del modelo

El modelo consiste de una lattice L, d dimensional tal que:

$$d \in \mathbb{Z}^2 \tag{3.1}$$

Sea A el conjunto finito de estados permitidos para una célula, en relación de las especies presa (roble escarlata, roble blanco) presentes en el modelo, tal que:

$$A = \{empty, early, late, mutant\} \tag{3.2}$$

Dónde:

- $early$ identifica un roble escarlata.
- $late$ identifica un roble blanco.
- $mutant$ identifica la especie mutante del roble blanco.
- $empty$ identifica una célula donde no se encuentra árbol alguno.

Además sea B, el conjunto finito de estados permitidos para una célula, que denota la presencia o ausencia de insectos en la misma i.e.

$$B = \{empty, full\} \tag{3.3}$$

De acuerdo a lo anterior es posible expresar a S (conjunto de estados) de la siguiente manera:

$$S = \{(a, b) \mid a \in A \wedge b \in B\} \tag{3.4}$$

La vecindad más recurrente en el modelo es la vecindad de Moore, la cual se define como:

$$N_{i,j}^{M} = \{(k, l) \in L \,||k - i| \leq r \wedge |l - j| \leq r\} \tag{3.5}$$

Se ha utilizado una frontera periódica para la simulación, en consecuencia es posible visualizar el ecosistema como un toroide.

3.1.1. Reglas del modelo

A partir de la descripción del ecosistema dada en la sección 2.4.3, es posible construir una secuencia de eventos compuesta de 10 reglas (Figura 23). Estas reglas modelan las interacciones que existen entre las especies presa y su depredador. Se considera transcurrida una temporada una vez que las 10 reglas han sido aplicadas. Las reglas son las siguientes:

1. **Mortalidad local M_l**. Si una célula está ocupada por un árbol de alguna especie, su probabilidad de cambiar este estado a $empty$ es proporcional al número de árboles que lo rodean de la misma especie. Esta regla utiliza la vecindad de Moore de radio 1. Denotemos la regla anterior de la manera siguiente:

$$C_{(i,j)_a}^{t} \neq empty \Rightarrow P\left(C_{(i,j)_a}^{t+1} = empty\right) = \eta k_{M_l} \tag{3.6}$$

Dónde:

- η denota el número de árboles de la misma especie que rodean a $C_{i,j}$
- k_{M_l} es el factor de mortalidad local, $0 \leq k_{M_l} \leq 12$.

2. **Mortalidad global.** Cada año, un árbol puede morir con probabilidad M_g. Esta regla modela la naturaleza orgánica de los individuos que componen el ecosistema, ya que estos pueden morir por diversas causas cada año. Esta regla esta expresada por:

$$C^t_{(i,j)_a} \neq empty \Rightarrow P\left(C^{t+1}_{(i,j)_a} = empty\right) = M_g \qquad (3.7)$$

Observe que esta regla no requiere información del estado de las células que pertenecen a la vecindad.

3. **Influjo de insectos.** Cada año una pequeña fracción ($I = 0.1$ %) de los arboles recibirá insectos. Esta propiedad fue agregada al modelo para garantizar la presencia de insectos durante toda la simulación.

$$C^t_{(i,j)_b} = empty \Rightarrow P\left(C^{t+1}_{(i,j)_b} = full\right) = 0.001 \qquad (3.8)$$

Esta regla al igual que la expresada en (3.7) no toma en cuenta el estado de las células de la vecindad.

4. **Mortalidad temprana de insectos.** Durante la primera parte de la temporada, los insectos solo pueden vivir en árboles de la especie temprana. Así que, si están en una célula sin árbol o con un árbol de la especie tardía (o su mutante) morirán. La regla se expresa de la manera siguiente:

$$C^t_{(i,j)_b} = full \wedge C^t_{(i,j)_a} \neq early \Rightarrow C^{t+1}_{(i,j)_b} = empty \qquad (3.9)$$

5. **Daño a robles de la especie temprana.** Si un árbol está ocupado por un insecto, su probabilidad de morir se incrementa. Cuando los insectos viven solo en árboles de la especie temprana la probabilidad de que estos mueran es del 35%.

$$C^t_{(i,j)_b} = full \wedge C^t_{(i,j)_a} = early \Rightarrow P(C^{t+1}_{(i,j)_a} = empty) = 0.35 \qquad (3.10)$$

6. **Migración**. Los insectos migran por difusión. Este fenómeno se modela a través de un AC de bloque. Los detalles esta regla se discuten en la sección 3.2.

7. **Mortalidad tardía de insectos.** Durante la segunda parte del año aquellos insectos en células vacías o con un árbol de la especie temprana morirán. Insectos en árboles de la especie tardía (o su mutante) sobrevivirán.

$$Si\ C^t_{(i,j)_b} = full \wedge C^t_{(i,j)_a} = early \Rightarrow C^{t+1}_{(i,j)_b} = empty \qquad (3.11)$$

8. **Daño a robles de la especie tardía.** En la segunda parte de la temporada, cuando los insectos viven solo en la especie tardía (o su mutante), la probabilidad de que estos mueran es más alta (igual a 1.0). Esta probabilidad es consecuencia del desarrollo de los insectos en las etapas anteriores, ya que el consumo per cápita se ha incrementado.

$$C^t_{(i,j)_b} = full \wedge C^t_{(i,j)_a} \neq early \Rightarrow P(C^{t+1}_{(i,j)_a} = empty) = 1 \qquad (3.12)$$

9. **Reproducción de árboles.** Sí una célula está vacía, se elige una de sus ocho vecinas al azar. La célula vacía cambiara su estado al estado de la vecina elegida. Si esta célula esta también vacía, la célula original permanecerá vacía.
10. **Reproducción de insectos.** Al final de un año, los insectos se reproducirán y se difundirán a un subconjunto de árboles en su vecindad. Insectos que se han estado alimentando de un árbol de la especie tardía durante la segunda parte de la temporada se reproducirán y difundirán al 50% de la vecindad de 5×5 (elegidos al azar), es decir $P_r(late) = 0.5$. Insectos en un árbol de la especie mutante se reproducirán y difundirán al 95% de esta vecindad, es decir, $P_r(mutant) = 0.95$.

Figura 23. Secuencia de eventos para el modelo propuesto.

3.2.Difusión

La dispersión del calor de una sustancia química en otra ha sido modelada a través de la ecuación de difusión de Navier – Stokes. En una dimensión esta ecuación toma la forma:

$$\frac{\partial u}{\partial t} = C\frac{\partial^2 u}{\partial x^2} \tag{3.13}$$

Donde $u(x,t)$ es la cantidad de calor o concentración de la sustancia que se dispersa a la distancia x en el tiempo t, y c es un constante que depende del medio. Una desventaja de esta solución es que la ecuación predice para una sustancia dada, alguna cantidad de la sustancia para cualquier instante de tiempo. Si se arroja una cierta cantidad de azúcar en una piscina, la ecuación

de difusión para este evento predice siempre una pequeña cantidad de azúcar lejos del sitio inicial de dispersión incluso cuando $t \to 0$.

Una alternativa para modelar la difusión es utilizar un AC, en donde una célula no blanca representa una partícula de azúcar y las células blancas representan el medio donde ocurre la difusión (Figura 24).

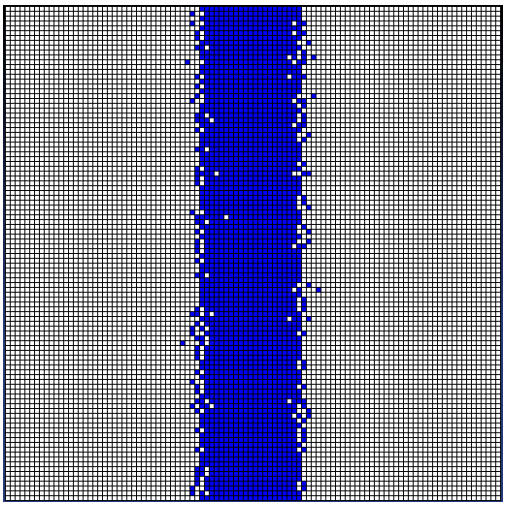


Figura 24. Modelo de difusión simple a través de un AC.

Para un modelo de difusión simple como el que se muestra en la Figura 24, la función de transición estipula que para cada fila que compone la lattice, se debe elegir de manera aleatoria una célula e intercambiar su estado con la célula adyacente a su izquierda. El procedimiento anterior constituye la primera evolución del autómata. Posteriormente el procedimiento se repite, sin embargo, la célula elegida intercambiara su estado con la célula adyacente a su derecha. De este modo en cada evolución impar del autómata, la célula elegida intercambia su estado con la célula adyacente a su izquierda, y en cada evolución par, lo intercambia con la célula adyacente a su derecha.

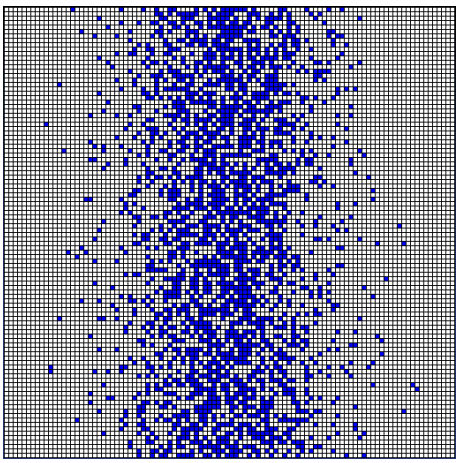


Figura 25. Evolución de la función de transición de un modelo de difusión simple.

Datos cuantitativos pueden ser extraídos del Autómata Celular al elegir una columna en particular, i.e. una distancia x a partir de la fuente de difusión, y sumar el número de células ocupadas n. Entonces el porcentaje $\frac{n}{N}$, donde N es el número de células en la columna, es un valor de densidad para $u(x,t)$ en el instante t, lo que representa la cantidad de azúcar presente a una distancia y tiempo dados

3.2.1. El modelo HPP

El primer modelo conocido como LGCA se debe a Hardy, de Passis, y Pomeau, este modelo es conocido como HPP. Es un modelo bidimensional de las interacciones de las partículas de un fluido. El modelo adopta una lattice rectangular, además, cada bloque de la lattice puede o no contener una partícula. Las características de cada partícula (masa y momento) son unitarias para simplificar la simulación.

En cada etapa de tiempo, una partícula viaja al nodo más cercano de acuerdo a su vector velocidad. Un principio de exclusión prohíbe que dos partículas ocupen la misma célula al mismo tiempo. Sin embargo, cada sitio de la lattice puede tener hasta 4 partículas. Debido a la geometría de la lattice las células solo pueden viajar en una de cuatro direcciones posibles.

Las reglas de interacción para partículas (Figura 26) que se encuentran en el mismo sitio son las siguientes:

a) Dos partículas viajando en direcciones opuestas al llegar a un bloque dan lugar a dos partículas viajando en ángulos rectos opuestos a las direcciones originales de desplazamiento.
b) Cualquier otra interacción provoca que las partículas conserven su dirección actual de desplazamiento.

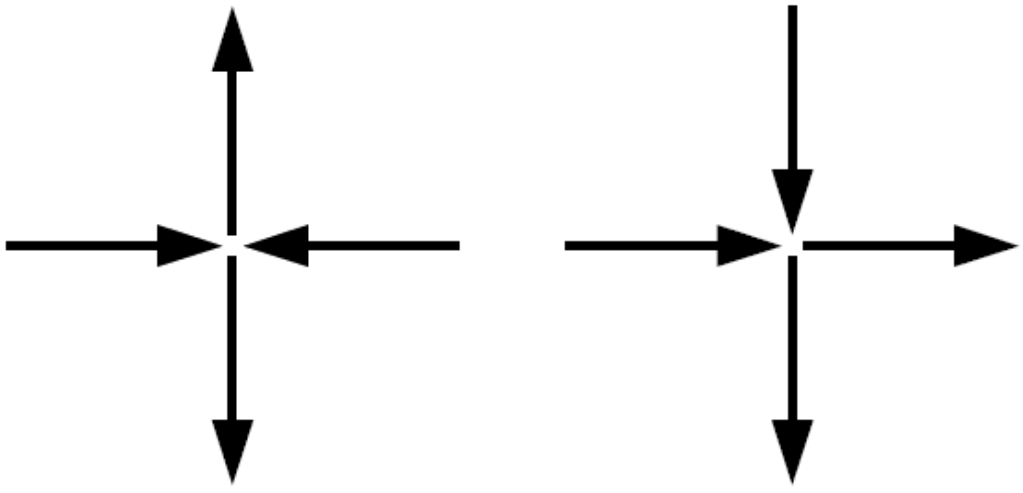


Figura 26. Interacción de partículas en el modelo HPP [4].

El modelo HPP puede ser modelado a través de un Autómata Celular de Bloque, un BCA (Block Cellular Automaton) está totalmente definido por:

$$(d, Q, V, O, t) \tag{3.14}$$

Dónde:

- d es la dimensión de la lattice, $d \in \mathbb{Z}^+$.
- Q es un conjunto finito de estados.
- $V = [0, v_1 - 1] \times [0, v_2 - 1] \times \ldots \times [0, v_d - 1]$ es un subconjunto de $\mathbb{Z}^d$. Un BCA particiona su lattice $\mathbb{Z}^d$ en bloques de tamaño $v_1 \times v_2 \times \ldots \times v_d$. Una partición es identificada por su origen $o^i \in \mathbb{Z}^d$
- $O = (o^i)_i$ es la secuencia finita de los origines o^i de las particiones utilizadas.
- $t: Q^w \to Q^w$, es la función de transición local de bloque.

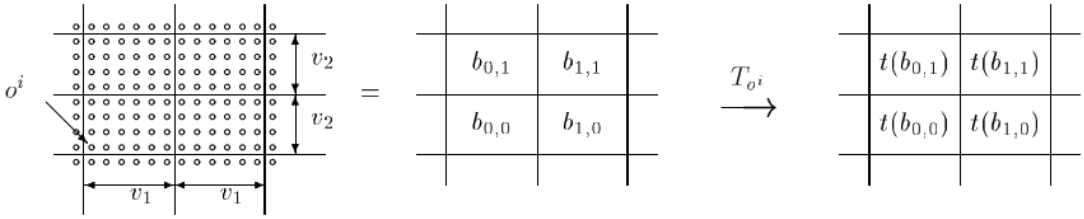


Figura 27. Actualización de bloques en un BCA [29].

De acuerdo a (3.14), el modelo HPP está totalmente definido por:

$$(2, \{0,1\}, (\{0,0\}, \{0,1\}, \{1,0\}\{1,1\}), O, t_{HPP}) \tag{3.15}$$

El esquema de particionamiento utilizado por este modelo es conocido como vecindad de Margolus, concepto introducido en [30]. Esta vecindad particiona la lattice en bloques de 2×2 celulas. En una evolución impar los bloques se mueven abajo y a la derecha en la lattice, en evoluciones pares los bloques regresan a su posición original.

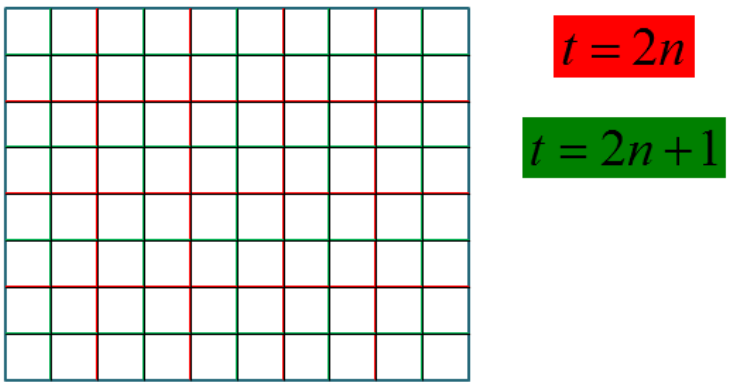


Figura 28. Vecindad de Margolus.

La función de transición local t_{HPP} es descrita utilizando notación de bloque de la manera siguiente:

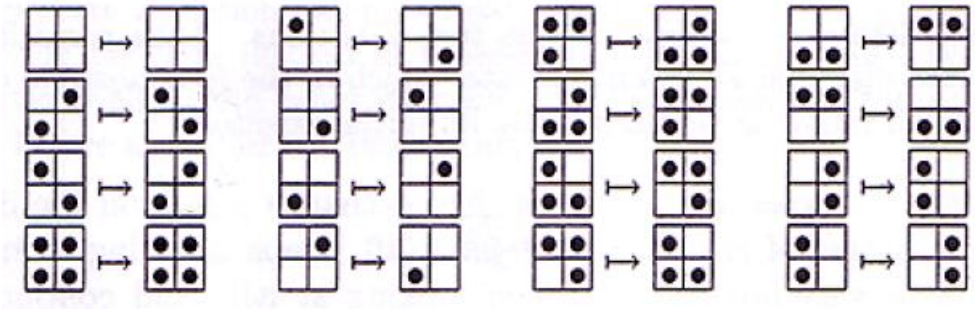


Figura 29. Notación de bloque que define la función de transición t_{HPP}.

Mediante t_{HPP} es posible modelar el movimiento libre de una partícula, así como colisiones entre las mismas (Figura 30). Note que las colisiones son perfectamente elásticas y que el número de partículas (que son idénticas en masa y velocidad) permanece constante en todo momento. El modelo de difusión resultante (Figura 31) a partir de esta regla captura una de las propiedades macroscópicas de la dinámica de fluidos: la conservación del momento.

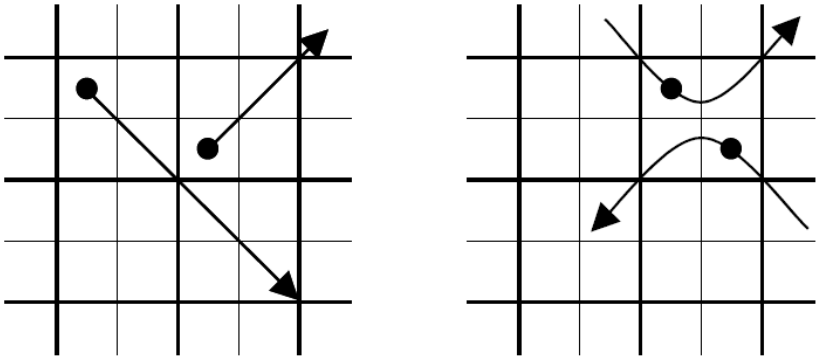


Figura 30. Movimiento libre de partículas y una colisión [4].

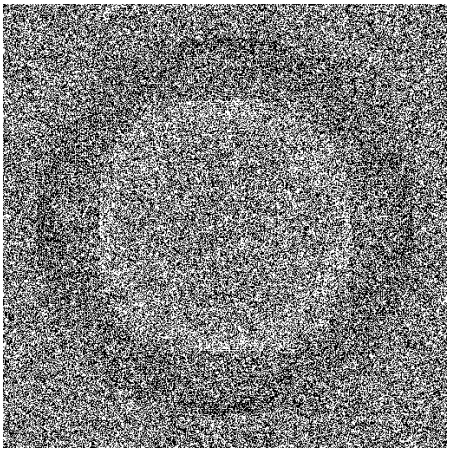


Figura 31. Evolución del modelo de difusión mediante un BCA.

3.3.Modelación de interacciones locales mediante redes de Petri

En la sección 2.2 se presentaron los conceptos fundamentales correspondientes a las redes de Petri, sin embargo, existen varias extensiones a esta técnica de modelado de sistemas, que resultan fundamentales para modelar las interacciones locales de un autómata celular. Varias de estas técnicas son discutidas y analizadas en [31].

3.3.1. Extensiones a la técnica de modelado con redes de Petri

A continuación se presentan las extensiones a la teoría de redes de Petri que serán utilizadas para modelar el sistema propuesto.

- **Arcos inhibidores**. Son arcos que van de un sitio p a una transición t; tal que, la transición t solo es habilitada si no existen marcas en p. Los arcos inhibidores permiten modelar la prohibición de una transición de estado. Son denotados por un arco con un círculo en uno de sus extremos (Figura 32).

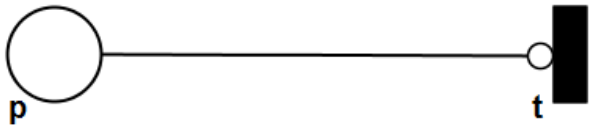


Figura 32. Arco inhibidor.

- **Arcos habilitadores.** Son arcos utilizados para verificar la presencia de marcas en un sitio de entrada. El disparo de la transición t asociada a este tipo de arcos no consume marcas del sitio de entrada p.

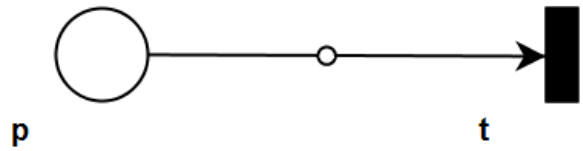
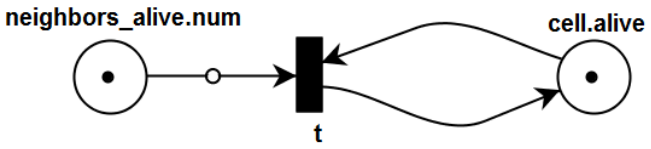


Figura 33. Arco habilitador.

- **Sitios con capacidad limitada.** Con esta extensión se limita el número de marcas que un sitio puede contener. De esta manera, una transición solo podrá disparar si genera a lo más un número de marcas tal que, no se exceda la capacidad del sitio de salida donde las marcas serán depositadas.
- **Marcas con atributos.** A través de esta extensión, individuos con diferentes atributos pueden pertenecer a un solo sitio. De esta manera, diferentes clases de individuos pueden ser representados por el marcado de un sitio. Al ser disparadas, las transiciones necesitan definir qué tipo de marcas generaran en base al tipo de marcas consumidas. La Figura 34 muestra una RP donde el sitio $cell.alive$ puede contener marcas que identifican células, que a su vez tienen el atributo $alive$, el cual puede tomar los valores verdadero y falso. El sitio $neighbors\_alive.num$

contiene una marca que representa el número de vecinos “vivos” en la vecindad de una célula. La transición t puede cambiar el atributo $alive$ de una marca del estado falso al verdadero. Note que el arco que une $neighbors\_alive.num$ con t es un arco habilitador, el cual solo verifica los atributos de las marcas contenidas en ese sitio.



$$neigbors.alive = 3 \wedge cell.alive = false \Rightarrow cell.alive = true$$

Figura 34. Marcas con atributos en una red de Petri.

- **Redes de Petri jerárquicas.** La capacidad para analizar un sistema de manera jerárquica es fundamental cuando se analizan sistemas complejos. En una RP secciones enteras que encapsulan cierta funcionalidad del sistema, pueden ser reemplazadas por un sitio o una transición en un nivel de abstracción más alto. De manera similar, sitios y transiciones pueden ser reemplazados por subredes, que provean un mayor detalle en cuanto a la funcionalidad que cada uno representa. La interface entre la red en un nivel de abstracción más alto y aquella en uno más bajo está definida por: los sitios de entrada y salida de la transición que se está detallando, o las transiciones que entregan o recogen marcas del sitio cuya funcionalidad se describe en un nivel de abstracción más bajo. En la Figura 35a tanto el sitio $P1$, como la transición $T3$, describen un comportamiento con un alto nivel de abstracción (aparecen sombreados en gris), de modo que, surge la necesidad en un momento dado de detallar el comportamiento que ambos describen (Figura 35b). Note además que en la especialización de la transición $T3$, aquellos sitios que son parte de la interfaz con la abstracción a alto nivel aparecen como círculos con un borde más grueso.

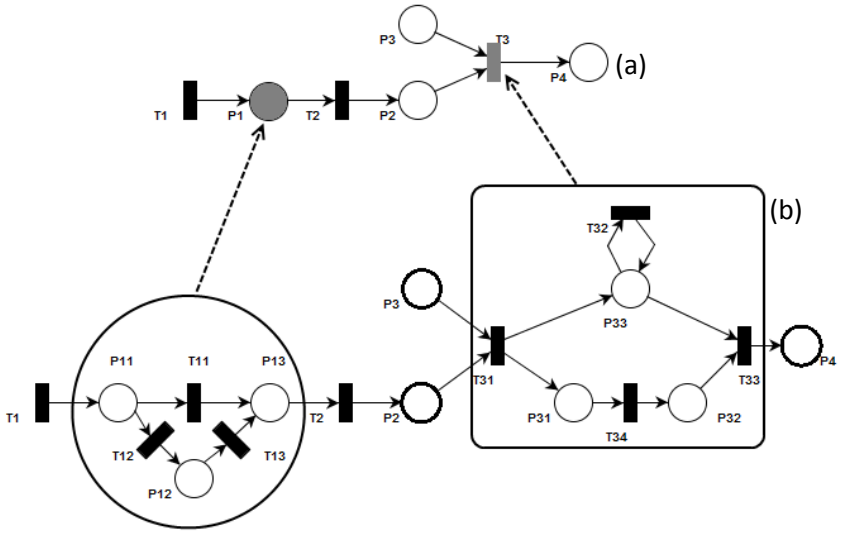


Figura 35. Red de Petri jerárquica (Adaptado de [8]).

3.3.2. Clasificación de reglas

Cada una de las reglas descritas en la sección 3.1.1 puede ser clasificada en tres categorías (Figura 36), de acuerdo al esquema de cómputo necesario para modelarla. Los esquemas de cómputo considerados son los siguientes:

- Autómatas finitos probabilísticos (AFP). En las reglas caracterizadas por este esquema, una célula no requiere más información que la propia para determinar su estado en $t + 1$ (ver sección 2.3).
- Autómatas celulares clásicos (AC). En estas reglas, una célula depende no solo de su propio estado, sino del estado de las células que se encuentran en su vecindad. Además, esta evolución procede de manera síncrona para todas las células en etapas discretas de tiempo.
- Autómatas celulares de bloque (BCA). En este esquema, se considera un particionamiento en bloques de la lattice. La función de transición del autómata se aplica de manera síncrona a cada bloque de la lattice.

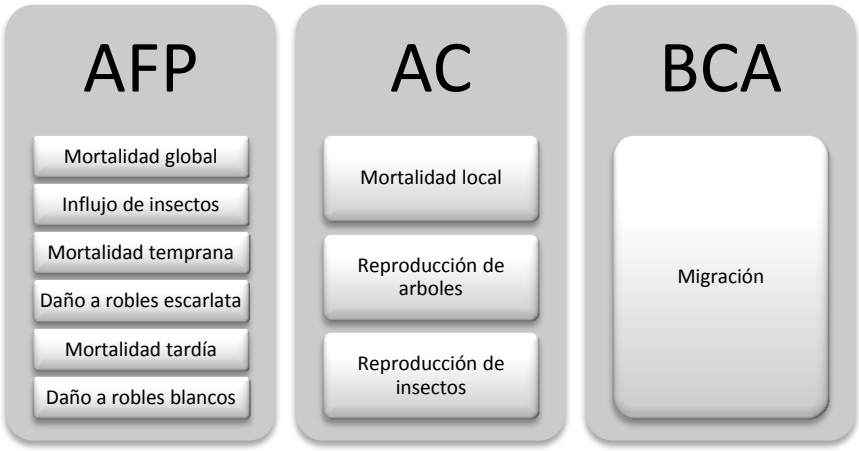


Figura 36. Clasificación de reglas del modelo propuesto.

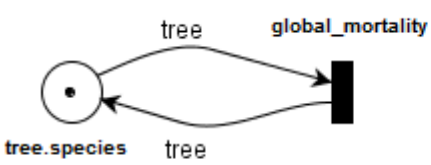
De manera independiente al esquema de cómputo que la representa, una regla se ocupa de la transición de estados de una célula. Cada célula en la lattice será modelada a través de dos sitios, en los cuales se distribuye la información relacionada a la especie de árbol que la habita, así como de la presencia o ausencia de un insecto en ella. Estos sitios son:

- $tree.species$. Este sitio contiene marcas que representan arboles con el atributo $especies$. El atributo $species$ puede tomar cualquier elemento del conjunto A descrito en la sección 3.1.
- $insect.is\_present$. Este sitio contiene marcas que representan insectos con el atributo $is\_present$, este atributo puede tomar los valores $full$ o $empty$, tal y como se describió en la sección 3.1.

En cada regla modelada, el cómputo de una transición aparece en un recuadro de pseudocódigo.

3.3.3. Mortalidad global

Cada temporada cierto número de árboles muere con probabilidad M_g. Una transición $global\_mortality$ cambia el estado del atributo $especies$ al valor $empty$ con probabilidad M_g (Figura 37).

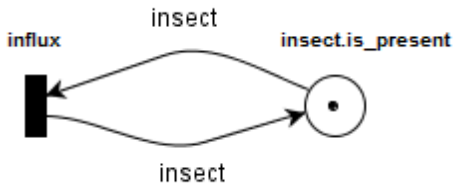



```
IF tree.species != empty
THEN
   tree.species <- empty WITH p <- Mg
ENDIF
```

Figura 37. Mortalidad global de árboles.

3.3.4. Influjo de insectos

Cada temporada el 0.1% de los arboles recibe insectos. Una transición $influx$ toma una marca del sitio $insect.is\_present$ y verifica el estado del atributo $is\_present$, de tener el valor $empty$, la transición cambiara este valor a $full$ con una probabilidad igual a 0.001 (Figura 38).

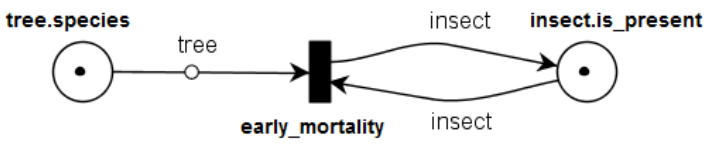



```
IF insect.is_present = empty
THEN
   insect.is_present <- full
   WITH p <- 0.001
ENDIF
```

Figura 38. Influjo de insectos

3.3.5. Mortalidad temprana de insectos

Durante su etapa de larvas los insectos solo pueden sobrevivir en un árbol de la especie temprana, en consecuencia, insectos que se encuentren en un árbol de la especie tardía morirán. Para modelar esta regla, es necesaria una transición $early\_mortality$, la cual a través de un arco habilitador verifica el estado de una marca del sitio $tree.species$, al mismo tiempo, esta transición obtiene la marca del sitio $insect.is\_present$, y verifica el estado del atributo $is\_present$. Si el atributo $species$ de la marca $tree$ no tiene el valor $early$, entonces de ser necesario, al atributo $is\_present$ de la marca $insect$ le es dado el valor $empty$ (Figura 39).

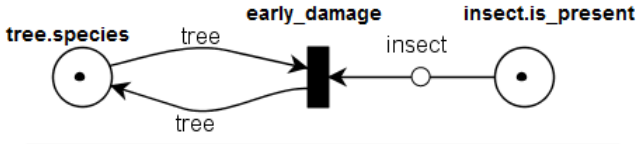



```
IF tree.species != early
THEN
   IF insect.is_present = full
   THEN
      insect.is_present <- empty
   ENDIF
ENDIF
```

Figura 39. Mortalidad temprana de insectos.

3.3.6. Daño a robles de la especie temprana

En la Figura 40 una transición $early\_damage$ consumirá una marca $tree$ del sitio $tree.species$, además esta transición verificará el estado $is\_present$ de la marca $insect$. Si el atributo $especies$ de la marca $tree$ analizada tiene el valor $early$, y el atributo $is\_present$ tiene el valor $full$, entonces el árbol analizado tiene una probabilidad de morir igual a 0. 35%, en caso de que el árbol muera el atributo $species$ tomará el valor $empty$.

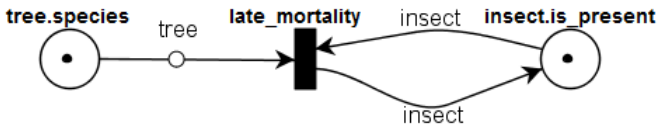



```
IF tree.species = early AND insect.is_present = full
THEN
   tree.species <- empty
   WITH p <- 0.35
ENDIF
```

Figura 40. Daño a robles de la especie temprana.

3.3.7. Mortalidad tardía de insectos

En su etapa adulta los insectos solo pueden alimentarse de árboles de la especie tardía. Debido a esto, cualquier insecto en su etapa adulta que se encuentre en un árbol de la especie temprana morirá. La modelación de esta regla es análoga a la regla "Mortalidad temprana de insectos", la Figura 41 muestra la red de Petri correspondiente a esta regla.




```
IF tree.species != late
THEN
   IF insect.is_present = full
   THEN
      insect.is_present <- empty
   ENDIF
ENDIF
```

Figura 41. Mortalidad tardía de insectos

3.3.8. Daño a robles de la especie tardía

En su etapa adulta los insectos tienen un mayor consumo de recursos per cápita, en consecuencia cualquier árbol de la especie tardía que se encuentre poblado por un insecto morirá. La RP que modela esta regla (Figura 42) es análoga a la discutida en la regla de "daño a robles escarlatas".

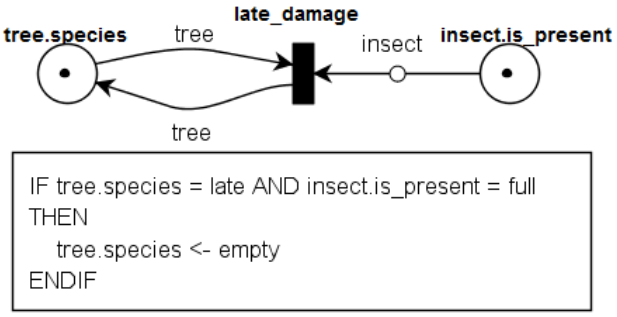


Figura 42. Daño a robles de la especie tardía.

3.3.9. Secuencialidad de reglas

A través de los sitios $tree.species$ e $insect.is\_present$, es posible unir las RPs de la reglas modeladas en las secciones 3.3.3 a 3.3.8. Sin embargo, las transiciones podrían ser habilitadas de manera concurrente, y el comportamiento de la red construida seria no determinista.

Para dar secuencialidad al comportamiento de la red se han introducido sitios especiales denotados por números. Cada transición consume una marca de estos sitios, y genera una marca en un sitio de salida de la misma naturaleza. En la Figura 43 podemos observar que solo existe una marca en el sitio inicial 01, de modo que, los eventos en la red proceden de manera secuencial.

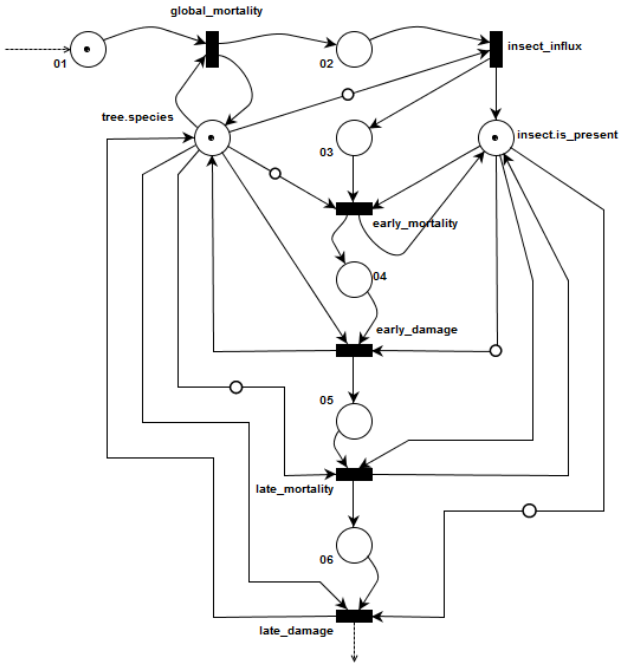


Figura 43. Red con secuencialidad.

3.3.10. Mortalidad local

A diferencia de las reglas en las secciones 3.3.3 a 3.3.8, en la mortalidad local, una célula interacciona con las células de su vecindad para determinar su estado en $t + 1$. Las reglas que incluyen este tipo de interacciones utilizan la vecindad de Moore con $r = 1$ o 2. Denotaremos a

las células en la vecindad de C, como C_x, $x \in [1,8]$. En la Figura 44 se puede observar esta convención.

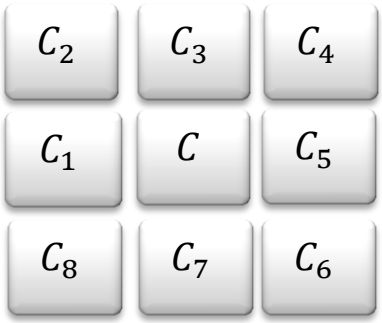


Figura 44. Una célula y su vecindad.

Al inicio de cada temporada los árboles mueren debido a los efectos de la competencia intraespecífica e interespecífica (ver secciones 4.1 y 4.2). La probabilidad de que un árbol muera depende del número de individuos que lo rodea (así como de su especie), una transición: $local\_mortality$ toma una marca $tree$ del sitio $tree.species$ y se comunica con las células en su vecindad, sitios denominados $tree.species\_x$, $x \in [1,8]$ (en el caso de la vecindad de Moore de radio 1) representan a los árboles en cada una de esas células. Una vez realizado el conteo en las células vecinas, la transición $local\_mortality$ determina el destino del árbol analizado, en caso de morir, la transición cambia el atributo $species$ de la marca $tree$ analizada al valor $empty$. En la Figura 45, se muestra el comportamiento anterior, observe además el uso de los sitios numerados, la regla de mortalidad local representa el inicio de los eventos que una célula experimenta a lo largo de una temporada. Dado que existe una marca en el sitio 00, la transición está habilitada, tras su disparo, esta deposita una marca en el sitio 01, lo que inicia la secuencia de eventos en la RP de la Figura 43.

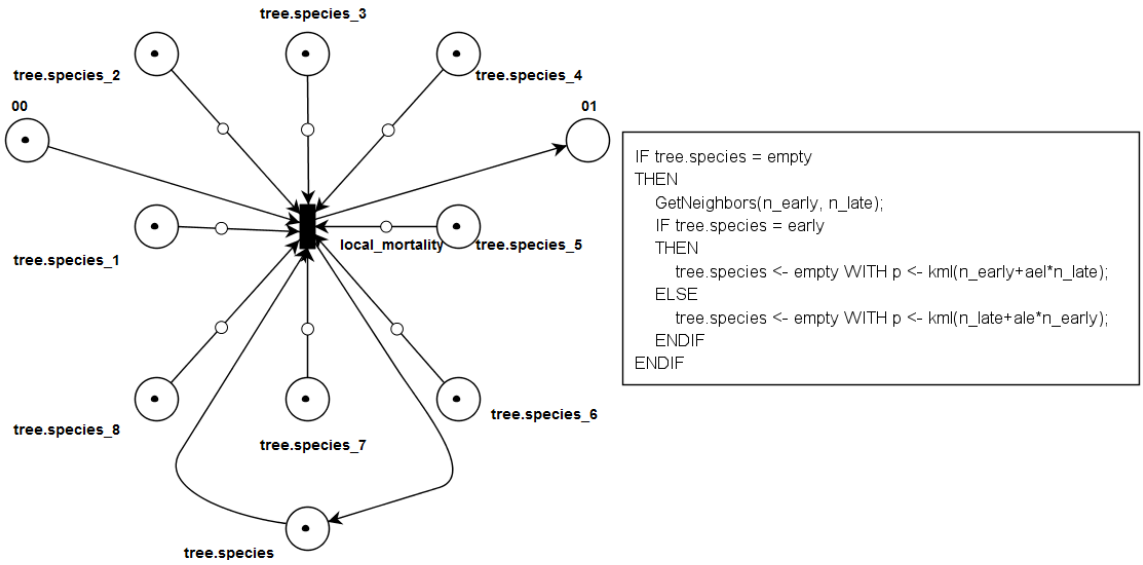


Figura 45. Mortalidad local.

3.3.11. Migración

Hasta ahora la unidad funcional del modelo ha sido la "célula", y es en base a esta unidad funcional que todas las reglas han sido modeladas, sin embargo, dado que la migración de los insectos es simulada a través de un autómata celular de bloque, es necesario modelar la migración considerando como unidad funcional a cada uno de los bloques que integran la lattice.

Ya que la migración involucra únicamente a los insectos, solo es necesario tomar en cuenta los sitios $insect.is\_present$, para modelar un bloque. Dado que el tamaño v de los bloques en el modelo HPP es igual a 2 (ver sección 3.2.1), cada bloque se compone de 4 células. Además haremos uso de una transición HPP que consumirá las marcas de los sitios de cada célula, y de acuerdo a la función de transición del modelo HPP realizará los cambios de estado correspondientes. En la Figura 46 se puede observar la RP utilizada para modelar un bloque de acuerdo a las ideas expuestas.

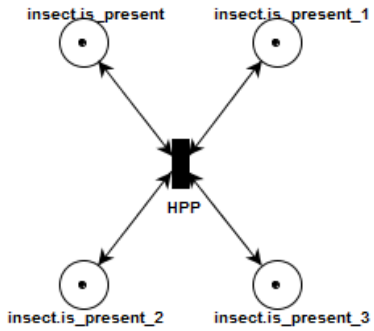


Figura 46. Un bloque de un BCA modelado a través de una RP.

Según el modelo propuesto, la difusión es un proceso activo durante cinco generaciones, por lo que es necesario contar con un mecanismo de regulación que, una vez transcurrido este tiempo detenga la difusión y, active las reglas siguientes del modelo. Con este fin se hará uso del patrón de diseño "*Remove all*" expuesto en [32]. En la Figura 47 se observa la red de Petri de este patrón de diseño. Después de que una marca ha sido puesta en el sitio $enable\_removal$, la transición $remove$ puede disparar. Una marca $enable$ es puesta en este sitio con cada disparo de la transición $remove$, en consecuencia esta transición puede disparar repetidamente hasta que, no haya más marcas en el sitio $placeToBeEmptied$. Cuando esto ocurre el arco inhibidor activa la transición $endRemoval$, la cual consume la marca del sitio $enable\_removal$ y produce una marca en el sitio $done$, en este punto el proceso finaliza.

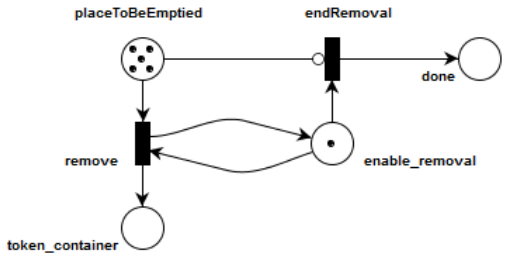


Figura 47. Patrón de diseño "Remove all".

El proceso de difusión en un BCA se logra a través de dos esquemas de particionamiento que se alternan en el tiempo (Figura 28). Observe que no importando cual sea la partición activa (par o impar), un bloque es autosuficiente, ya que las células contenidas en el no requieren información alguna de las células contenidas en otros bloques. Sin embargo, cuando la partición de la lattice cambia, también lo hace el bloque al que una célula pertenece, y por lo tanto también cambian las células con las que interactúa. En la Figura 48 se puede observar una lattice hipotética en la que se muestran los sitios $insect.is\_present$ correspondientes a 16 células; los bloques correspondientes a la partición par se encuentran separados por líneas rojas, de manera similar líneas azules separan los bloques de la partición impar. Por simplicidad en la lattice en la Figura 48, solo se muestra un bloque de la partición impar. Los sitios $insect.is\_present\_5$, 6, 9 y 10 son parte del mismo bloque en la partición impar, sin embargo, en la partición par cada una de ellos pertenece a un bloque distinto.

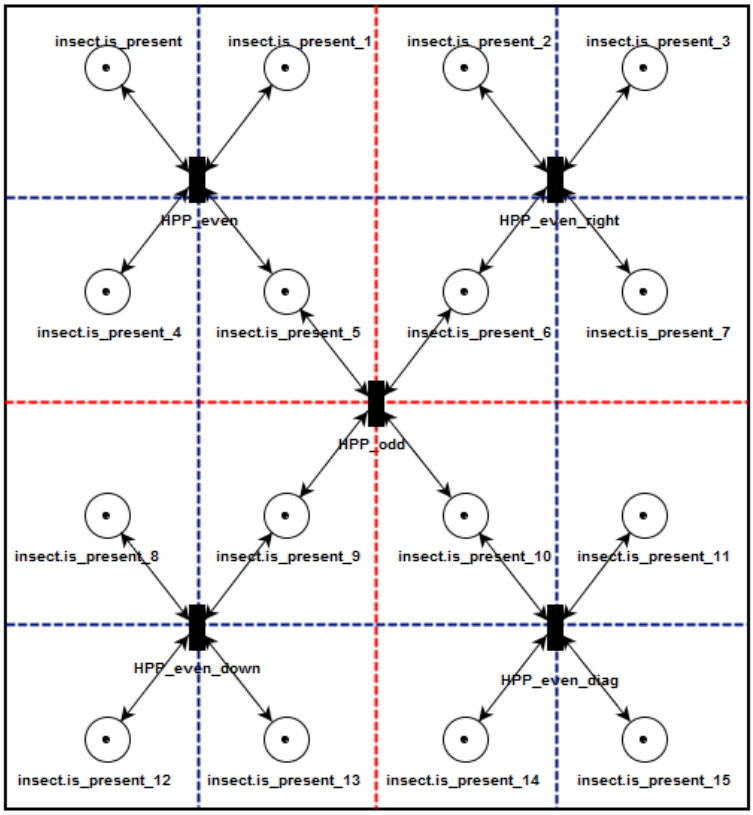


Figura 48. Interacción entre células en bloques pares e impares.

Ya que la función de transición en un BCA es aplicada de manera síncrona a todos los bloques de la lattice, es necesario sincronizar las transiciones activas en una partición. Por ejemplo, en la RP de la Figura 48, las transiciones $HPP\_even, HPP\_even\_right, HPP\_even\_down$ y HPP_even_diag deben disparar de manera síncrona cuando la partición par está activa. Además es necesaria una interface que permita controlar el funcionamiento de cada bloque a través de la RP "Remove all". A continuación se describe los elementos para controlar y sincronizar el comportamiento de cada bloque.

- Sitios $enable$. Para sincronizar el disparo de las transiciones de cada bloque, se agregará un sitio de entrada $enable$ a cada una de ellas, el marcado inicial de estos sitios es cero.
- Transición $even\_enable$. Los sitios $enable$ de cada bloque, serán sitios de salida de esta transición. De este modo es posible sincronizar el disparo de cada transición y en consecuencia, el comportamiento de cada bloque.
- Sitio $done$. Este sitio forma parte de la RP del patrón de diseño "Remove all" (Figura 47). Al estar conectado a través de un arco inhibidor con las transiciones de cada bloque, este sitio deshabilitara el disparo de las transiciones tan pronto una marca sea depositada en él, de este modo cuando transcurran cinco generaciones el proceso de migración termina.
- Sitio $even\_done$. Es un sitio de salida para cada transición HPP_even en cada bloque, de manera que sincroniza la función de transición en la partición par. Además es un sitio de entrada para la transición $reactivate$.
- Transición $reactivate$. Esta transición tiene como único sitio de entrada al sitio $even\_enable$, de modo que, se habilita cuando todas las transiciones en la partición par hayan disparado. La tarea principal de esta transición es reactivar la RP "Remove all", en consecuencia el sitio $enable\_removal$ será un sitio de salida para esta transición. Finalmente la transición $reactivate$ tiene como sitio de salida al sitio $odd\_partition$, cuyo comportamiento es descrito a continuación.

En la Figura 49, se observa la RP de un bloque donde se incluyen los mecanismos de control mencionados.

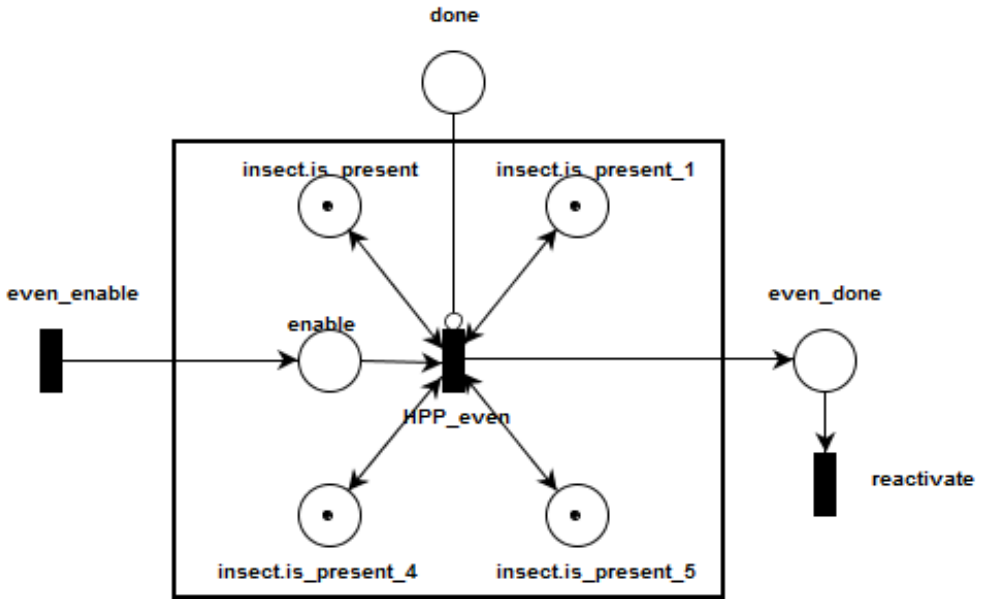


Figura 49. RP para un bloque y mecanismos de control.

Para modelar el cambio de partición se introducirá un sitio $odd\_partition$, con un marcado inicial de cero, indicando que la partición impar está inactiva. Este sitio, es además un sitio de entrada para la transición HPP_odd de la Figura 48, de modo que, cuando exista una marca en este sitio, las transiciones de la partición impar sean habilitadas. Dado que cuando la partición impar está inactiva, las transiciones de la partición par deben ser deshabilitadas, un arco inhibidor conecta el sitio $odd\_partition$ a la transición $even\_enable$, de este modo, se impide el disparo de esta transición cuando la partición impar está activa. La transición $reactivate$ tiene como sitio de salida a $odd\_partition$, de este modo, cuando $reactivate$ dispare, las transiciones de la partición impar son activadas. La Figura 50 muestra la RP que modela la etapa de migración en el modelo propuesto, de acuerdo a las ideas expuestas en esta sección.

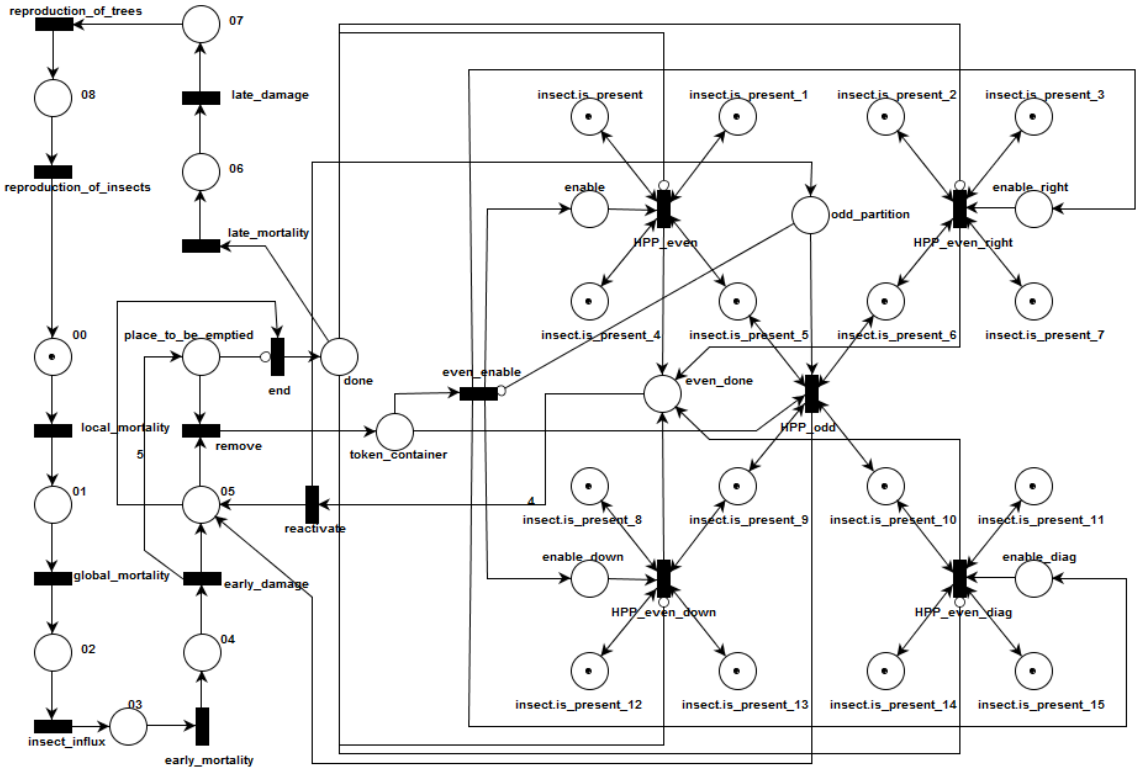


Figura 50. Migración

3.3.12. Reproducción de arboles

Cada temporada una célula no poblada por un árbol, cambia su estado de manera no determinista al estado que tiene alguna de las células en su vecindad; e. g. si la vecina elegida está poblada por un roble blanco, tras la aplicación de esta regla, la célula bajo análisis también estará poblada por un roble blanco. La RP que describe el comportamiento de esta regla es análoga a la RP de la Figura 45, la única diferencia es el evento de la transición, el cual elige al azar una de las vecinas, y cambia el estado de la célula al de la vecina elegida. La Figura 51 muestra la RP que modela la regla de reproducción de árboles.

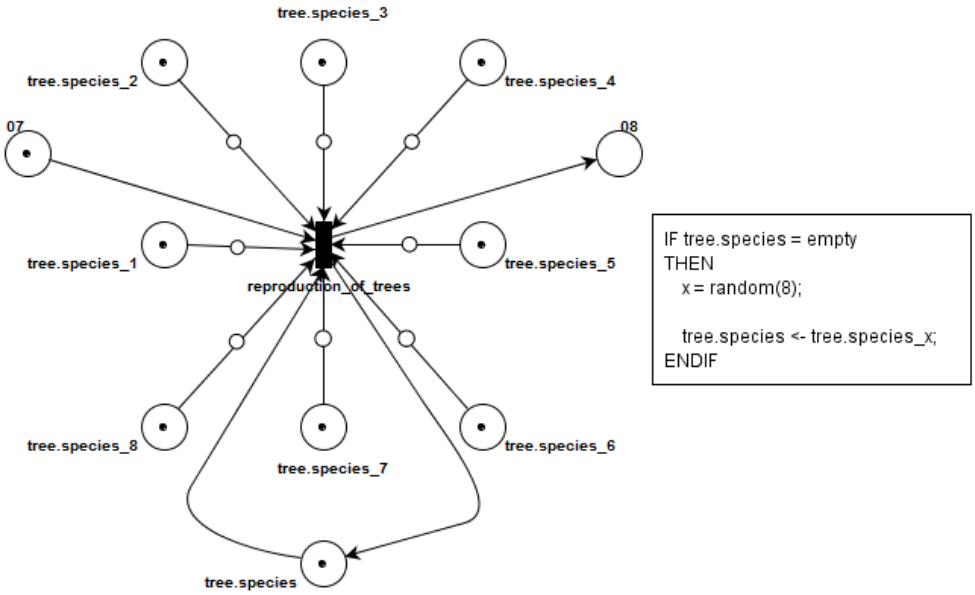


Figura 51 Reproducción de árboles.

3.3.13. Reproducción de insectos

Al final de cada temporada, un insecto se reproduce de manera aleatoria a un número de células en su vecindad de Moore de radio 2. La RP que modela la reproducción de insectos es análoga a las redes que modelan las reglas de "Mortalidad local" y "Reproducción de árboles", note que, a diferencia de estas reglas la transición de esta red deposita marcas en los sitios de la vecindad. En la Figura 52 se muestra la RP que modela la reproducción de insectos.

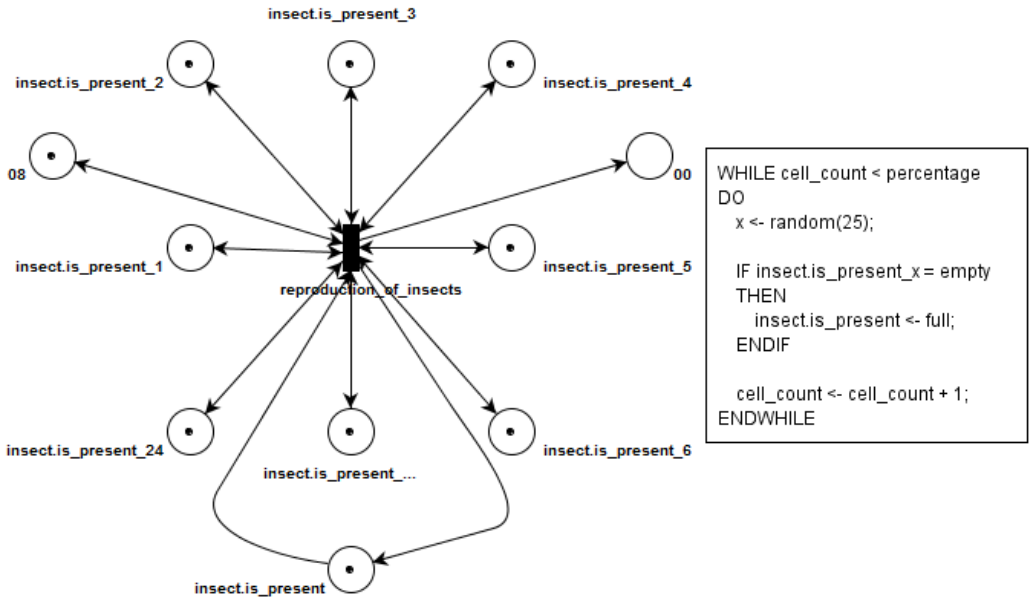


Figura 52. Reproducción de insectos.

3.4.Implementación del modelo de depredación secuencial

Dado el extenso número de reglas que componen al modelo y la gran cantidad de operaciones de lectura / escritura que se dan en la vecindad de una célula, es necesaria una implementación del modelo tal que, sea posible alcanzar un desempeño razonable en la simulación del mismo. No se debe perder de vista el hecho de que se está simulando un modelo de computo paralelo (un modelo basado en ACs) utilizando procesadores secuenciales.

Con el objetivo de mantener la implementación del modelo libre de los detalles técnicos (lenguaje de programación, plataforma de software, etc.) de un sistema operativo en particular, se presentan las ideas detrás de la implementación con un alto nivel de abstracción.

3.4.1. Implementación de la lattice

La lattice es implementada utilizando un mapa de bits en formato BGR24 (Figura 53), tomando ventaja de la geometría rectangular de la lattice, es posible considerar cada pixel del mapa de bits como una célula. Además la lectura y escritura de un pixel son procesos rápidos utilizando métodos de acceso directo a memoria.

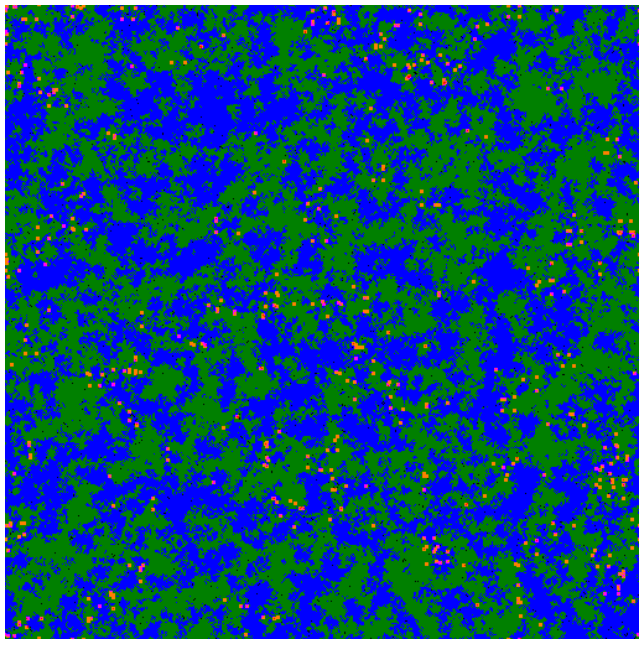


Figura 53. Lattice implementada utilizando un mapa de bits.

3.4.2. Implementación de los estados

En el modelo de depredación secuencial, cada célula contiene información acerca de la especie de árbol que la habita, así como de la presencia o ausencia de insectos en la misma. Para almacenar esta información en cada pixel del mapa de bits, es necesario distribuirla en las componentes de color del pixel. En el formato de color BGR24 cada pixel destina 8 bits para las componentes B (Blue – Azul), G (Green – Verde) y R (Red – Rojo), las diferentes combinaciones de los valores de estas componentes se traducen en diferentes colores (Figura 54).

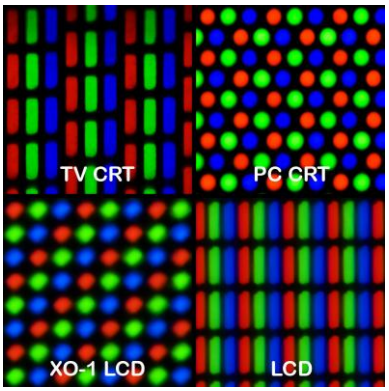


Figura 54. Pixeles en diferentes dispositivos.

De acuerdo a lo anterior tenemos 24 bits de almacenamiento distribuidos en tres componentes de 8 bits. La distribución de la información en cada pixel se muestra en la Tabla 1.

Tabla 1. Distribución de la información de una célula dentro de un pixel.

Organismo	Componente	Valor de Ausencia	Valor de Presencia
Árbol de la especie temprana	B	0	255
Árbol de la especie tardía	G	0	255
Árbol mutante de la especie tardía	G	0	128
Insecto	R	0	255

La distribución propuesta permite representar de manera natural una célula del ecosistema modelado, habitada por una cierta especie de árbol y / o insectos. Por ejemplo, un pixel correspondiente a una célula que no está habitada por árboles o insectos tendrá sus tres componentes con un valor de cero, y en consecuencia presentará un color negro; de manera similar una célula habitada por un árbol de la especie temprana, pero no por insectos, presentará un color azul, finalmente un árbol mutante de la especie tardía habitado por insectos tendrá un color naranja. La Tabla 2 resume los colores que representan los estados que un pixel puede adquirir a lo largo de una simulación.

Tabla 2. Colores que representan los estados de una célula.

Color	Organismos presentes en una célula			
	Árbol de la especie temprana	Árbol de la especie tardía	Árbol mutante de la especie tardía	Insectos
Negro				
Azul	X			
Verde oscuro		X		
Verde			X	
Magenta	X			X
Naranja			X	X
Amarillo		X		X

Note que los árboles de la especie tardía (normal y mutante) comparten la componente G de un pixel, hacerlo de este modo evita recurrir a un formato de mapa de bits más grande, e. g. BGRA32. Además, esta representación resulta consistente, ya que ambos arboles solo difieren debido a una mutación, y al ser de la misma especie son representados con distintas tonalidades de verde.

Algunas de las ventajas de esta implementación son:

- Al representar una célula directamente a través de un pixel, cualquier cambio en el estado de la misma se verá reflejado inmediatamente en la lattice, lo que elimina la necesidad de métodos de actualización de estados y, permite al mismo tiempo alcanzar un mejor desempeño en la simulación.
- Dado que la mayoría de las reglas del modelo actúan sobre una especie en específico en una etapa de tiempo dada, sólo es necesario leer una de las componentes para verificar si la célula reúne las condiciones necesarias para la aplicación de la regla. De haber designado un color para representar un estado, sería necesario leer las tres componentes para verificar que la célula está en un estado adecuado.
- El cálculo de las densidades de población de las especies presentes en el modelo, es fácil como consecuencia directa del punto anterior.
- Ya que la información de una célula es parte misma del mapa de bits, no es necesario crear variables adicionales para almacenarla, lo que reduce drásticamente el uso en memoria de la implementación.

Capitulo IV Resultados

A continuación se presentan las simulaciones realizadas con el modelo propuesto. En cada una se incluyen las condiciones propuestas para cada simulación, así como un análisis de los resultados obtenidos. Todas las simulaciones se realizaron utilizando el software CASimulator en una PC con las siguientes características:

- Procesador AMD Athlon™ X2 Dual – Core QL – 65, 2100 MHz.
- Cache L1 (código) 64 KB.
- Cache L1 (datos) 64 KB.
- Cache L2 (1 MB, 512 – 512).
- 2813 MB de memoria RAM física.
- Tarjeta gráfica ATI Radeon ™ HD 3200 Graphics.
- Sistema Operativo Windows 7 ™ Home Premium.
- Microsoft .NET Framework ™ versión 3.5 SP1.

4.1.Competencia intraespecífica

4.1.1. Introducción

Todos los organismos crecen, se reproducen y mueren. En el transcurso de su vida son afectados por las condiciones en las que viven, y por los recursos que son capaces de obtener. Sin embargo, ningún organismo vive en aislamiento, en al menos un periodo corto de su vida, ese organismo es miembro de una población compuesta de individuos de su misma especie.

Los miembros de una especie requieren de recursos similares para sobrevivir, sin embargo, su demanda combinada puede exceder los recursos disponibles. En consecuencia los miembros de una población compiten por los recursos disponibles, y en el transcurso de esta disputa algunos son privados del acceso a estos recursos.

La competencia es una interacción entre individuos, consecuencia de la necesidad de acceder a un recurso en común, que conlleva a una disminución en la probabilidad de supervivencia, crecimiento y / o reproducción de al menos uno de los organismos competidores. [1]

En el caso de la competencia intraespecífica las interacciones se dan entre individuos de la misma especie.

A continuación se analizan los efectos de la competencia intraespecífica en el crecimiento y mortalidad de una población de robles (especie temprana).

4.1.2. Crecimiento de una población bajo competencia intraespecífica.

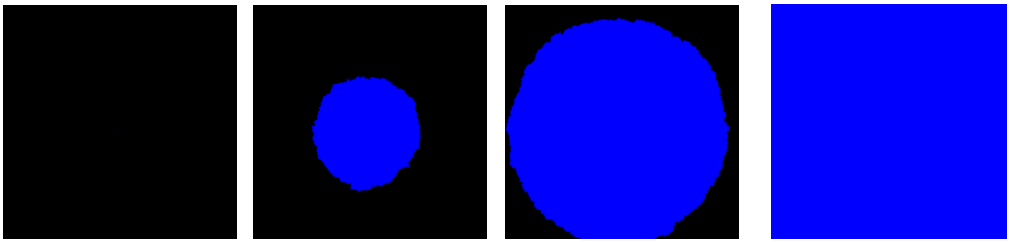
Los parámetros de la simulación son los siguientes:

- Factor de mortalidad local = 0
- Porcentaje de mortalidad global = 0
- Influjo de insectos = 0

- Tamaño de la lattice 512×512 células.
- Fronteras periódicas deshabilitadas.
- Número de generaciones simuladas = 9270

Observe que aquellos fenómenos que ocasionan la muerte de los individuos de una especie han sido deshabilitados (al poner sus parámetros a 0); esto con el fin de aislar los efectos de la competencia intraespecífica en el crecimiento de una población.

Esta simulación comienza con un solo individuo de la especie temprana de robles. La Figura 55 muestra la situación del ecosistema en $t = 0, t = 3000$, $t = 6000$ y $t = 9270$ generaciones. Dado que no existe la mortalidad de los individuos, la totalidad de la lattice ha sido cubierta por individuos de la especie temprana de robles (Figura 55d).



(a) (b) (c) (d)

Figura 55. Situación del ecosistema en $(a)\ t = 0$ $(b)\ t = 3000$ $(c)\ t = 6000$ $(c)\ t = 9270$

Sea N_t el tamaño de la población de alguna especie en t. El tamaño donde $N_t = N_{t+1}$ se conoce como capacidad de carga (K) y, representa el tamaño de la población que el ecosistema es capaz de sostener sin presentar una tendencia a crecer o decrecer. En este sentido se dice que K representa un equilibrio estable [1].

En el caso de esta simulación, $K =$ tamaño de la lattice ($512 \times 512 = 262144$ células), pues una vez que la población ha alcanzado este tamaño tenemos que $N_t = N_{t+1}$ (Figura 56). Observe sin embargo, que esta equivalencia solo es válida bajo las condiciones de aislamiento propuestas para esta simulación. Cuando en el ecosistema actúen además la competencia interespecífica y la depredación, la dinámica del crecimiento de una población será más compleja.

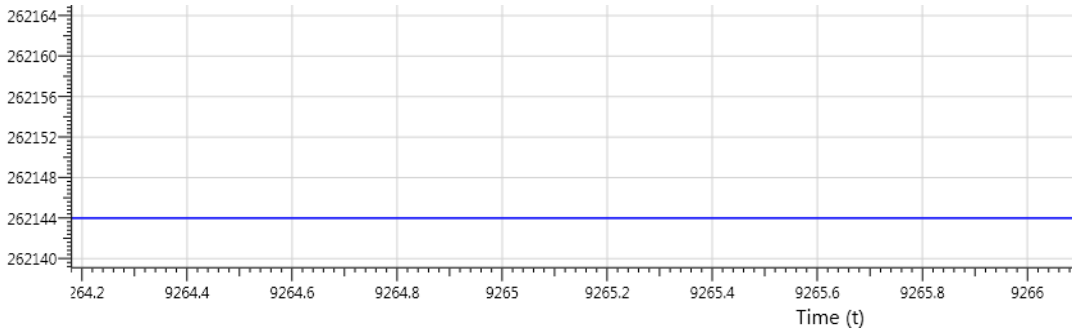


Figura 56. Equilibrio estable en $N_t = K$.

Se ha dicho que la competencia entre individuos es consecuencia de la necesidad compartida por un recurso, así que: ¿Cuál es el recurso por el que compiten los robles? En un principio la respuesta podría no ser aparente. Cuando la necesidad por un recurso es compartida, un competidor puede negar el acceso a este recurso (de manera deliberada o no) a otros competidores. En el caso de los robles, al ocupar una célula, estos establecen una zona de influencia alrededor de ellos, en la cual es factible que puedan reproducirse. Sin embargo, si los espacios disponibles en esta zona de influencia ya han sido ocupados por otros robles, la capacidad de reproducción que un roble pueda tener se verá disminuida. La Figura 57 muestra este comportamiento.

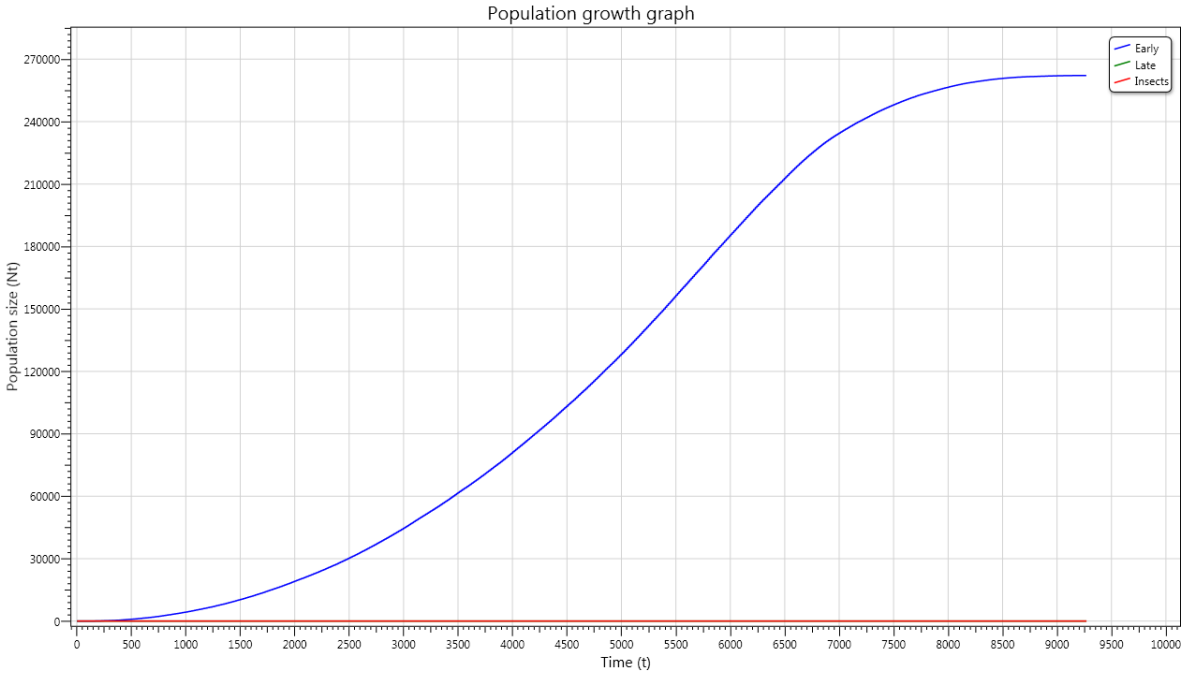


Figura 57. Dinámica del crecimiento de la población bajo los efectos de la competencia intraespecífica.

Cuando los individuos de una especie se introducen en una nueva área, generalmente lo hacen en un número reducido, esto tiene las siguientes consecuencias:

a) La cantidad de espacio per cápita disponible es muy grande, y en consecuencia existe un bajo nivel de competencia intraespecífica.
b) A pesar de lo anterior, el tamaño de la población crece lentamente (Figura 58a), ya que el número de individuos que se reproduce es bajo.

Conforme el tamaño de la población crece, también lo hará el nivel de competencia intraespecífica presente en la población. Sin embargo, los efectos negativos de la competencia intraespecífica en la capacidad reproductiva de la población son despreciables, en comparación del aporte que los nuevos individuos de cada generación hacen. Esto se refleja en un incremento cada vez mayor de N_t (Figura 58b). Este comportamiento se sostiene hasta que el nivel de

competencia intraespecífica es demasiado grande (cuando la disponibilidad per cápita de espacio es muy pequeña), en este punto la capacidad reproductiva de la población comienza a declinar, los que se traduce en un crecimiento cada vez más reducido de N_t (Figura 58c), hasta alcanzar la capacidad de carga del ecosistema.

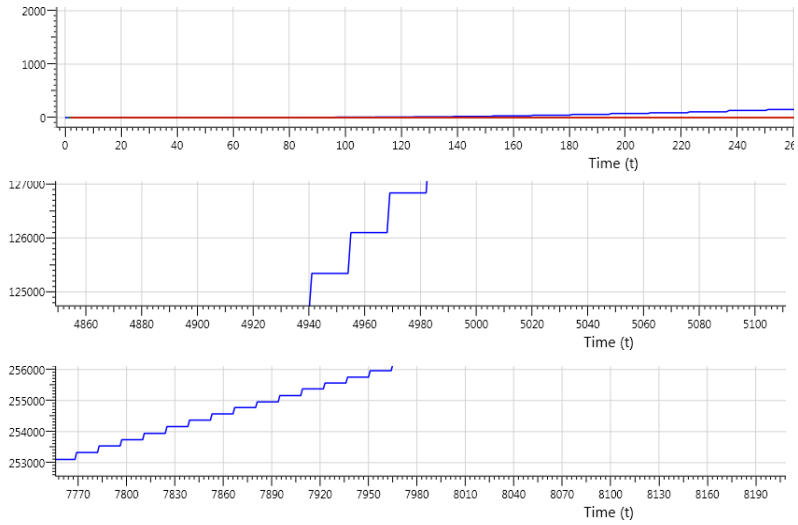


Figura 58. Variación en el tiempo de N_t.

4.1.3. Mortalidad de una población bajo competencia intraespecífica

Los parámetros de la simulación son los siguientes:

- Factor de mortalidad local = 1
- Porcentaje de mortalidad global = 0
- Influjo de insectos = 0
- Fronteras periódicas deshabilitadas.
- Número de generaciones simuladas = 10000

Además de afectar el crecimiento de una población, la competencia intraespecífica afecta la mortalidad de la misma (entendiendo por mortalidad el número de individuos que muera cada generación). El espacio no solo constituye un recurso necesario para la reproducción, sino que, es el contenedor de recursos subyacentes para que los robles puedan sobrevivir (nutrientes, agua, etc.). De modo que, cuando un roble ocupa una célula de la lattice, priva de manera efectiva a otros robles de los recursos presentes en esa porción de espacio. En consecuencia entre mayor sea el nivel de competencia intraespecífica, habrá menos probabilidades de que un roble sobreviva a la siguiente generación.

En cada generación nuevos individuos se incorporan a la población, sin embargo, por las razones antes expuestas, un cierto número de individuos morirá. El reclutamiento neto Δ_t denota el número neto de individuos que se incorpora a la población en cada generación, es decir:

$$\Delta_t = Número\ de\ nacimientos - Número\ de\ muertes \quad (4.1)$$

La Figura 59 muestra el comportamiento del ecosistema bajo las condiciones marcadas para esta simulación.

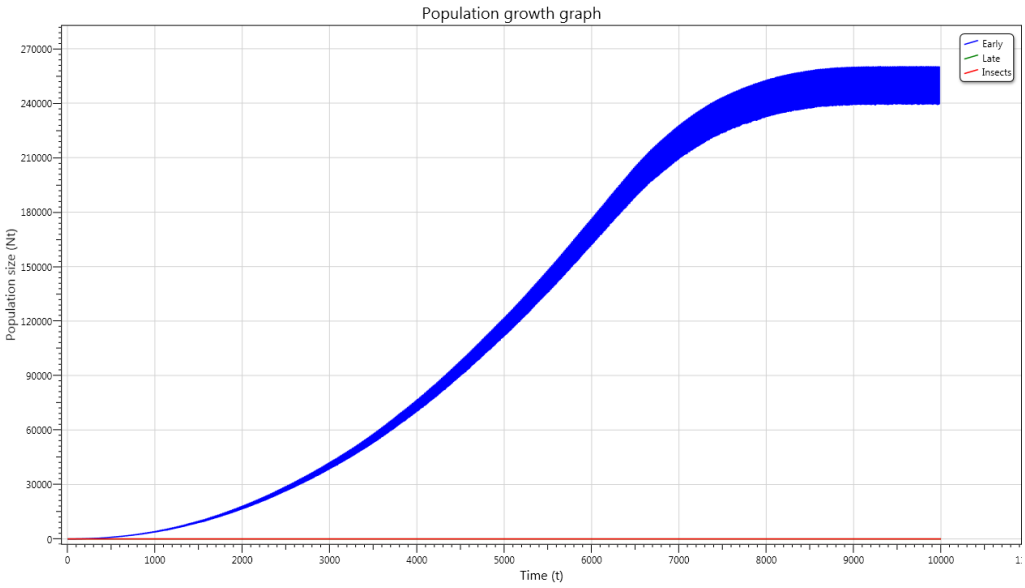


Figura 59. Crecimiento y mortalidad de una población bajo competencia intraespecífica.

Al inicio de la simulación N_t es pequeño, y como se analizó en 4.1.2, existe un bajo nivel de competencia intraespecífica y un número reducido de nacimientos cada temporada. Sin embargo, aunque el nivel de competencia es bajo, algunos individuos mueren cada temporada. Al igual que en la simulación anterior, el número de nacimientos sobre – compensa los efectos de la competencia intraespecífica, lo que se traduce en un creciente Δ_t (Figura 60).

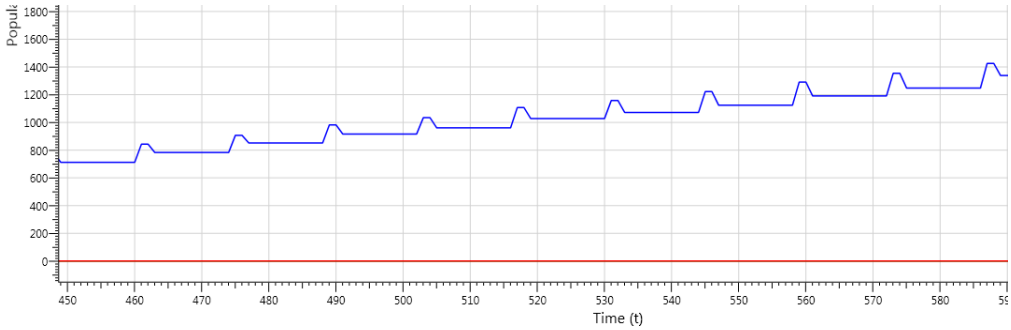


Figura 60. Dinámica de la población al inicio de la simulación.

Conforme N_t crece, existe mayor sobrecompensación entre el número de nacimientos y el número de muertes, lo que provoca que Δ_t crezca con mayor rapidez (Figura 61), hasta alcanzar un punto máximo. Después de este punto el nivel de competencia intraespecífica es tan grande que, dependiendo del número de muertes cada temporada, puede existir sobre o sub – compensación del número de nacimientos; esto se refleja en un Δ_t positivo o negativo según corresponda.

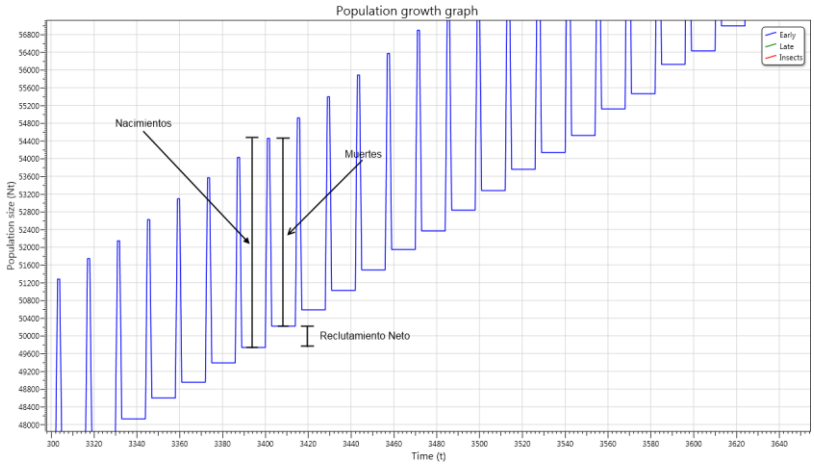


Figura 61. Comportamiento de Δ_t

Después de que Δ_t ha alcanzado su máximo, N_t entra en equilibrio asintótico estable, ya que las variaciones ocasionadas por los nacimientos o muertes no impiden que el sistema regrese a un valor acotado por una asíntota (Figura 62).

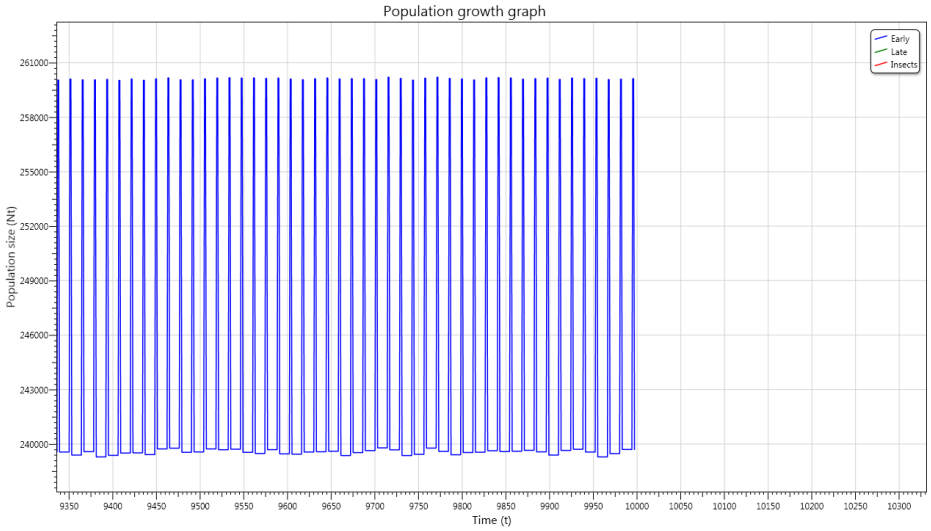


Figura 62. Equilibrio asintótico estable de una población.

4.2.Competencia interespecífica

4.2.1. Introducción

Los individuos de una población, no solo compiten con miembros de su propia especie, además, sufren de la explotación de recursos o interferencia por parte de individuos de otras especies. Este tipo de competencia es conocida como competencia interespecífica.

Un ejemplo clásico de competencia interespecífica aparece en [33]. En este trabajo se realizó un estudio de laboratorio utilizando tres especies de paramecios (organismos protozoarios). Estas especies fueron alimentadas con bacterias o células de levadura. Cuando las

especies fueron cultivadas en aislamiento se reprodujeron hasta alcanzar una capacidad de carga estable. Sin embargo, cuando la variedad p. aurelia y la variedad p. caudatum fueron cultivadas al mismo tiempo, los individuos de la especie p. caudatum siempre declinaron hasta extinguirse. En contraste cuando la variedad p. caudatum y la variedad p. bursaria se cultivaron juntas, ninguna variedad declinaba hasta el punto de la extinción, sino que, eran capaces de coexistir; sin embargo, la capacidad de carga de cada población era menor que si hubiera sido cultivada en aislamiento.

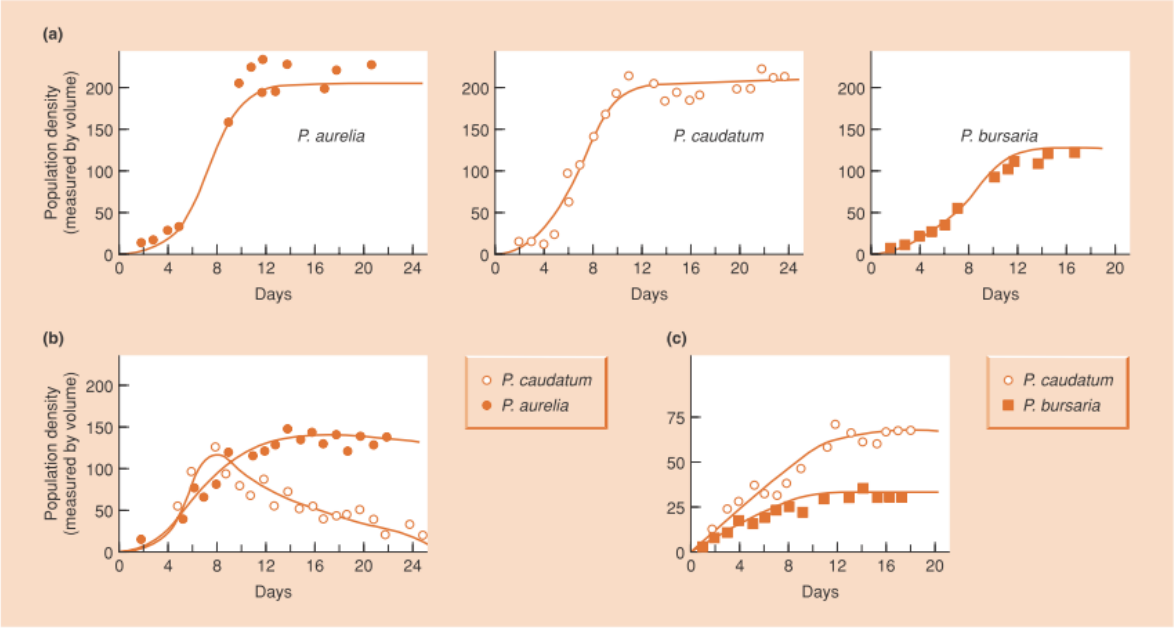


Figura 63. Competencia entre paramecios (a) p. aurelia, p. caudatum y p. bursaria al crecer en aislamiento alcanzan capacidades de carga estables. (b) Cuando crecen juntas la especie P. Aurelia lleva a la especie P. caudatum a la extinción (c) Las especies P. caudatum y P. bursaria son capaces de coexistir a una capacidad de carga menor [1].

La importancia de este experimento es que, se observan los diferentes resultados que la dinámica de la competencia interespecífica puede tener. Por un lado, vemos que los efectos de la competencia interespecífica son asimétricos. Como resultado de la competencia interespecífica, la especie p. aurelia lleva a la extinción a la variedad p. caudatum, mientras que el único efecto discernible para la variedad p. aurelia es una reducción en su capacidad de carga. Incluso cuando dos variedades son capaces de coexistir (el caso de p. caudatum y p. bursaria), una presenta una reducción mayor en su capacidad de carga en relación a su competidora.

4.2.2. Dinámica de la competencia interespecífica

Para determinar cuáles son las condiciones generales que permiten la coexistencia de dos especies competidoras, es necesario recurrir a los modelos matemáticos. En el modelo propuesto, durante la etapa de mortalidad local, la probabilidad de que un árbol muera está dada por:

$$P\left(C_{(i,j)_a}^{t+1} = empty\right) = k_{ml}\eta \qquad (4.2)$$

Dónde:

- k_{ml} es el factor de mortalidad local con $0 \leq k_{ml} \leq 0.12$
- η denota el número de árboles de la misma especie que $C^t_{(i,j)_a}$, ya que la mortalidad local opera con una vecindad de Moore de radio 1, tenemos que: $0 \leq \eta \leq 8$

La ecuación 4.2 regula la mortalidad de un árbol sujeto solo a competencia intraespecífica, es decir, la probabilidad de que un árbol muera es proporcional al número de individuos de la misma especie que se encuentran en la vecindad de $C^t_{(i,j)_a}$. Además de la competencia que ejercen los miembros de su propia especie, un árbol está sujeto a los efectos inhibitorios producto de la competencia con miembros de otra especie; por lo que es necesario incorporar a la ecuación 4.2 un término que contemple los efectos de la competencia interespecífica.

Sean η_{early} y η_{late} el número de árboles de las especies temprana y tardía respectivamente, en la vecindad de $C^t_{(i,j)_a}$. Suponga ahora que 10 árboles de la especie tardía tienen el mismo efecto inhibitorio en la especie temprana que un solo miembro de esta especie. El efecto total de la competencia (la probabilidad de que muera) será entonteces proporcional a $k_{ml}\left(\eta_{early} + \frac{1}{10}\eta_{late}\right)$. La constante $\frac{1}{10}$ es llamada coeficiente de competencia y es denotada por α_{e-l}. Este coeficiente mide el efecto competitivo per cápita que ejerce la especie tardía sobre la especie temprana. De este modo el producto $\alpha_{e-l}\eta_{late}$ representa una conversión de individuos de la especie tardía a individuos de la especie temprana. Observe que:

- $\alpha_{e-l} < 1$ significa que los individuos de la especie tardía tienen un efecto inhibitorio menor en los individuos de la especie temprana, que los individuos de la especie temprana sobre sí mismos.
- $\alpha_{e-l} > 1$ significa que los individuos de la especie tardía tienen un efecto inhibitorio mayor en los individuos de la especie temprana, que los individuos de la especie temprana sobre sí mismos.

De acuerdo a lo anterior podemos reescribir la ecuación 4.2 para la especie temprana como:

$$P\left(C^{t+1}_{(i,j)_a} = empty\right) = k_{ml}\left(\eta_{early} + \alpha_{e-l}\eta_{late}\right) \qquad (4.3)$$

Y en el caso de la especie tardía:

$$P\left(C^{t+1}_{(i,j)_a} = empty\right) = k_{ml}\left(\eta_{late} + \alpha_{l-e}\eta_{early}\right) \qquad (4.4)$$

Las ecuaciones 4.3 y 4.4 regulan la mortalidad de los árboles en un ecosistema bajo competencia intraespecífica e interespecífica.

4.2.3. Exclusión

Los parámetros de la simulación son los siguientes:

- Factor de mortalidad local = 1
- Porcentaje de mortalidad global = 0
- Influjo de insectos = 0
- Fronteras periódicas habilitadas
- Número de generaciones simuladas = 5000
- $\alpha_{e-l} = 1.5$
- $\alpha_{l-e} = 1$

La exclusión se da cuando un competidor fuerte lleva a una especie competidora a la extinción. En el ejemplo de los paramecios, la variedad aurelia lleva a la extinción a la variedad caudatum, los individuos de la variedad aurelia son entonces competidores fuertes.

Para simular una competencia excluyente se han elegido los valores de α_{e-l} y α_{l-e} de modo tal que, una especie sea un competidor fuerte en relación a su competidora. Observe que, de acuerdo a los valores elegidos la especie tardía será el competidor fuerte y llevará a la extinción a la especie temprana. Este comportamiento se observa en la Figura 64.

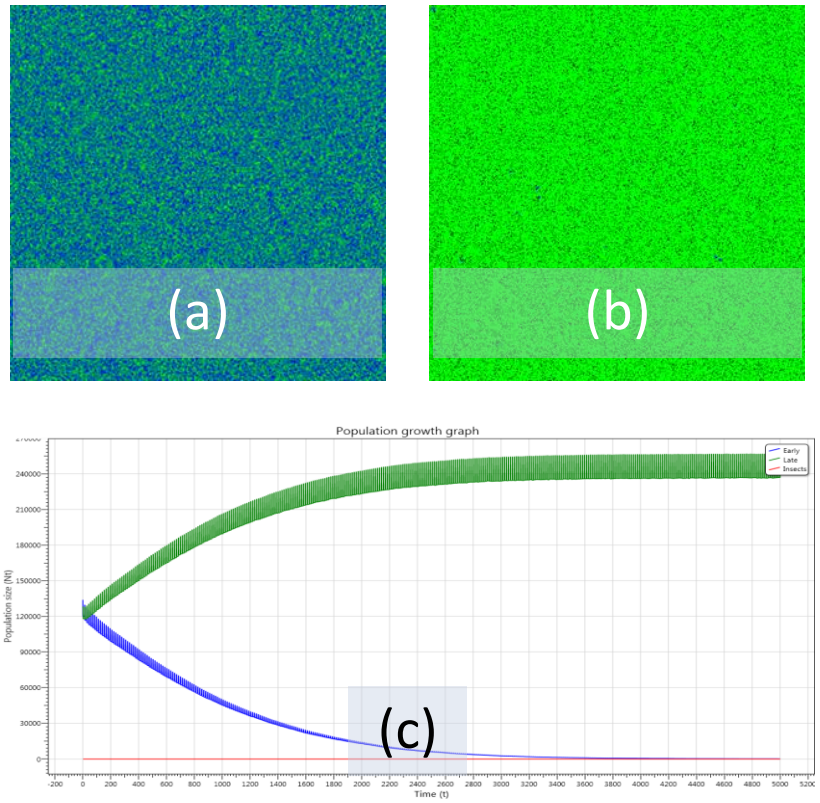


Figura 64. Competencia excluyente. (a) $t = 0$ (b) $t = 5000$, observe que la especie temprana casi se ha extinguido. (c) Dinámica poblacional.

4.2.4. Coexistencia

Los parámetros de la simulación son los siguientes:

- Factor de mortalidad local = 1
- Porcentaje de mortalidad global = 0
- Influjo de insectos = 0
- Fronteras periódicas habilitadas
- Número de generaciones simuladas = 10000
- α_{e-l} = 0.25
- $\alpha_{l-e} = 0.5$

En el ejemplo de los paramecios, la variedad caudatum y la variedad bursaria son capaces de coexistir. Sin embargo, los efectos de la competencia se reflejan en una reducción de la capacidad de carga de cada especie. Cuando una especie compite en menor grado con su especie competidora, que consigo misma, ambas especies son capaces de coexistir.

Los valores que se han elegido para α_{e-l} y α_{l-e} tienen dos propósitos:

a) Simular condiciones de coexistencia. Al ser α_{e-l} y α_{l-e} menores que uno, cada especie compite más con su propia especie, que con la especie competidora.
b) Ya que $\alpha_{e-l} < \alpha_{l-e}$, la especie temprana es un competidor más fuerte, y en consecuencia debe alcanzar una capacidad de carga mayor que la especie tardía.

La Figura 65 muestra los resultados de esta simulación.

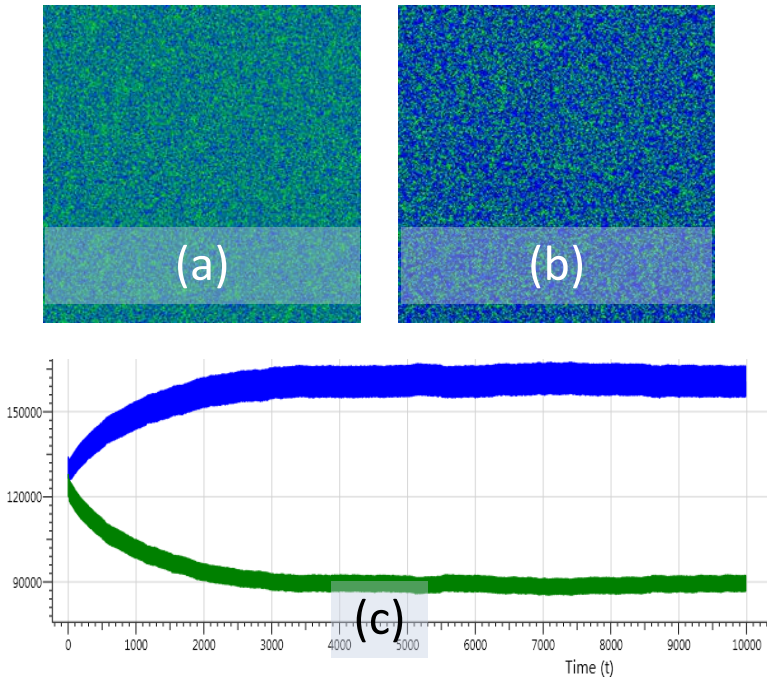


Figura 65. Coexistencia entre la especie temprana y la especia tardía. (a) $t = 0$ (b) $t = 10000$ (c) Dinámica poblacional.

4.2.5. Antagonismo mutuo

Los parámetros de la simulación son los siguientes:

- Factor de mortalidad local = 1
- Porcentaje de mortalidad global = 0
- Influjo de insectos = 0
- Fronteras periódicas habilitadas
- Número de generaciones simuladas = 10000
- Densidad inicial de la especie temprana = 60%
- Densidad inicial de la especia tardía = 40%
- $\alpha_{e-l} = 2$
- $\alpha_{l-e} = 2$

El antagonismo mutuo es una interacción entre dos especies donde, los individuos de cada especie compiten más con los individuos de su especie competidora que entre sí mismos. Interacciones de este tipo se dan cuando por ejemplo, cada especie tiene un comportamiento alelopático con su competidora, o cuando existe depredación mutua. El resultado de este comportamiento depende en gran medida de la densidad inicial de cada especie, la especie que posea una ventaja inicial llevara a la extinción a su competidora.

La Figura 66 muestra la dinámica de las especies en condiciones de antagonismo mutuo. Debido a la ventaja inicial de la especie temprana (azul), el tamaño de la población de la especie tardía (verde) declina hasta la extinción (Figura 66a – e). El antagonismo mutuo es una interacción en donde no es posible predecir el comportamiento de un ecosistema solo en términos del nivel de competencia entre especies. Un ligero cambio en el balance de las densidades de las especies competidoras puede llevar a una de ellas a la extinción.

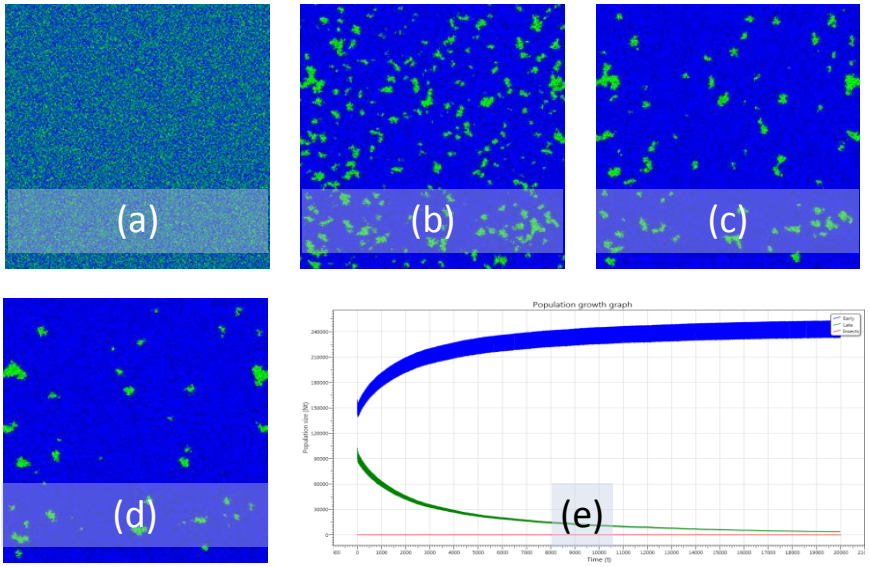


Figura 66. Antagonismo mutuo. (a) $t = 0$ (b) $t = 5000$ (c) $t = 10000$ (d) $t = 15000$ (e) Dinámica poblacional.

La Figura 67 muestra un fenómeno no observable con facilidad en modelos no espaciales, en ella se muestra la evolución de un ecosistema con una distribución inicial equitativa de ambas especies de robles (Figura 67a), con $\alpha_{e-l} = \alpha_{l-e} = 2.0$ y $k_{ml} = 1$. Conforme progresa la simulación comienza un proceso donde individuos de una misma especie se agrupan en regiones bien definidas Figura 67b. Dado que los individuos de ambas especies compiten en mayor medida con miembros de su especie competidora, la probabilidad de que un árbol muera dentro de un grupo compuesto por miembros de su misma especie, es menor que si entra en contacto con individuos de otra especie. De este modo cada individuo se ve beneficiado al formar parte de una comunidad de miembros de su propia especie, no solo porque dentro de esta comunidad tiene una mayor probabilidad de supervivencia, sino porque al formar grupos grandes tiene mayores posibilidades de exterminar a miembros de otras especies.

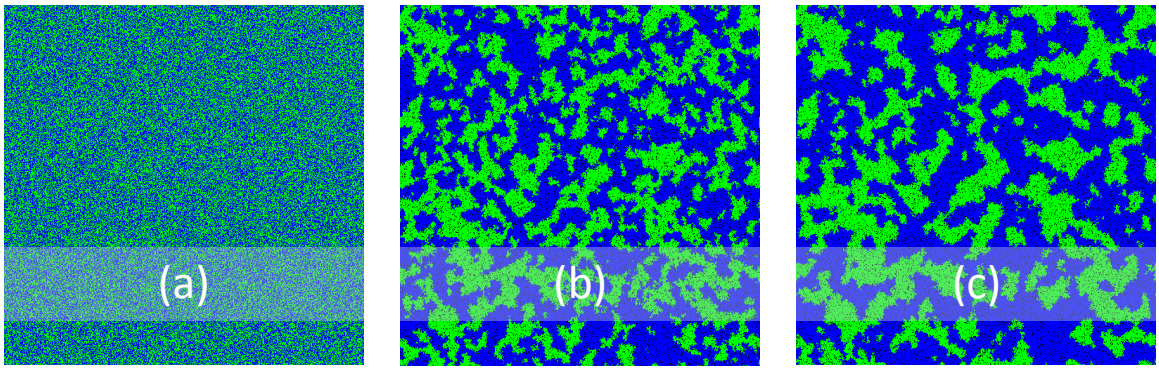


Figura 67. El antagonismo mutuo favorece la formación de comunidades de la misma especie.

4.3.Depredación secuencial

4.3.1. Introducción

La depredación secuencial es un caso de lo que Holt denominó "competencia aparente", en este tipo de interacciones, dos especies presa son afectadas por un depredador común. De este modo, cualquier incremento en la abundancia de la especie predadora, consecuencia del consumo de la presa 1, incrementa el daño recibido por la especie presa 2; de manera indirecta la especie presa 1 daña a la especie presa 2. En el caso de la depredación secuencial, la especie predadora afecta una especie presa distinta en diferentes etapas de su vida. De acuerdo a la sección 3.1 podemos resumir la dinámica de los insectos en las siguientes etapas:

a) Influjo de insectos.
b) Mortalidad temprana.
c) Daño a robles de la especie temprana.
d) Migración.
e) Mortalidad tardía.
f) Daño a robles de la especie tardía.

De esta dinámica podemos concluir lo siguiente:

- La competencia intraespecífica es un caso de competencia por interferencia, ya que durante la migración, los insectos bloquean la trayectoria de otros insectos, y en consecuencia existe competencia por el espacio.
- Ya que solo se tiene una especie predadora, no existe competencia interespecífica.

En contraste a los modelos clásicos, en el modelo propuesto, el crecimiento de la población de insectos no depende solo de la abundancia de las especies presa. Es decir, para que la población de insectos alcance una capacidad de carga estable, no basta con que exista un número abundante de especies presa. Lo anterior es consecuencia de la depredación secuencial, en su estudio de campo Futuyma y Wasserman observaron que los huevos de la alsophila pometaria eclosionaban al mismo tiempo que el follaje de la especie temprana estaba disponible. Sin embargo, las larvas eran incapaces de completar su desarrollo en follaje maduro, y por lo tanto, debían migrar a árboles de la especie tardía, los cuales ofrecían follaje joven del cual las larvas podían seguir alimentándose. Una migración exitosa garantiza que una larva alcanza la madurez y pueda reproducirse.

4.3.2. Migración

La migración de los insectos es modelada utilizando un autómata celular de bloque, ejecutando una regla de difusión, como el discutido en 3.2. Esto garantiza la conservación de la masa, es decir, el número de insectos que inicia la migración es igual al que la termina. Sin embargo, esto no significa que todos los insectos tienen una migración exitosa, ya que al final de la misma no todos los insectos se ubican en un árbol de la especie tardía, y en consecuencia algunos morirán durante la etapa de mortalidad tardía.

La probabilidad de que un insecto sobreviva a la migración, y pueda reproducirse depende de la distribución espacial de las especies temprana y tardía. Ya que solo se simulan 5 etapas de difusión, ningún insecto puede moverse más allá de 5 células del lugar donde inicia su migración. La consecuencia de esta limitación, es que una distribución asimétrica de árboles dificulta la migración de los insectos, regiones grandes compuestas de árboles de una sola especie actuaran como barreras naturales que limitarán la propagación de insectos en el ecosistema (Figura 68).

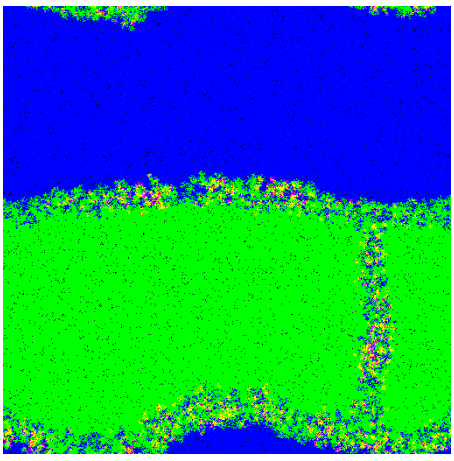


Figura 68. Los insectos se limitan a regiones donde existe una distribución mixta de árboles.

Aquí la noción de una región grande puede parecer ambigua, consideraremos una región grande como un conjunto de células tal que, al ser recorrida por un insecto existe al menos una trayectoria en la que existen cinco células consecutivas vacías o pobladas por la misma especie de roble. Para sobrevivir y reproducirse, los insectos necesitan regiones donde exista una alta probabilidad de tener una migración exitosa, es decir, regiones como esta representan los recursos necesarios para el desarrollo y la supervivencia de los insectos. Conforme los insectos se desarrollan, transforman el ecosistema a su alrededor y este recurso eventualmente se agota debido a su presencia, para sobrevivir los insectos deben encontrar nuevos recursos, de no haberlos la población de insectos puede extinguirse.

En la Figura 69 se observa la evolución de un ecosistema en donde los insectos consumen recursos. En la parte (a) se observa una distribución inicial aleatoria de 50% de individuos de la especie temprana y 50% de individuos de la especie tardía, en distribuciones como está existe un bajo número de "regiones grandes" y la cantidad de recursos disponibles para el desarrollo de los insectos es grande. Esta condición se refleja en un rápido crecimiento de la población de insectos, tal como se puede observar en la Figura 69c. Conforme los insectos consumen los recursos disponibles (Figura 69b), su población disminuye hasta alcanzar en una capacidad de carga estable.

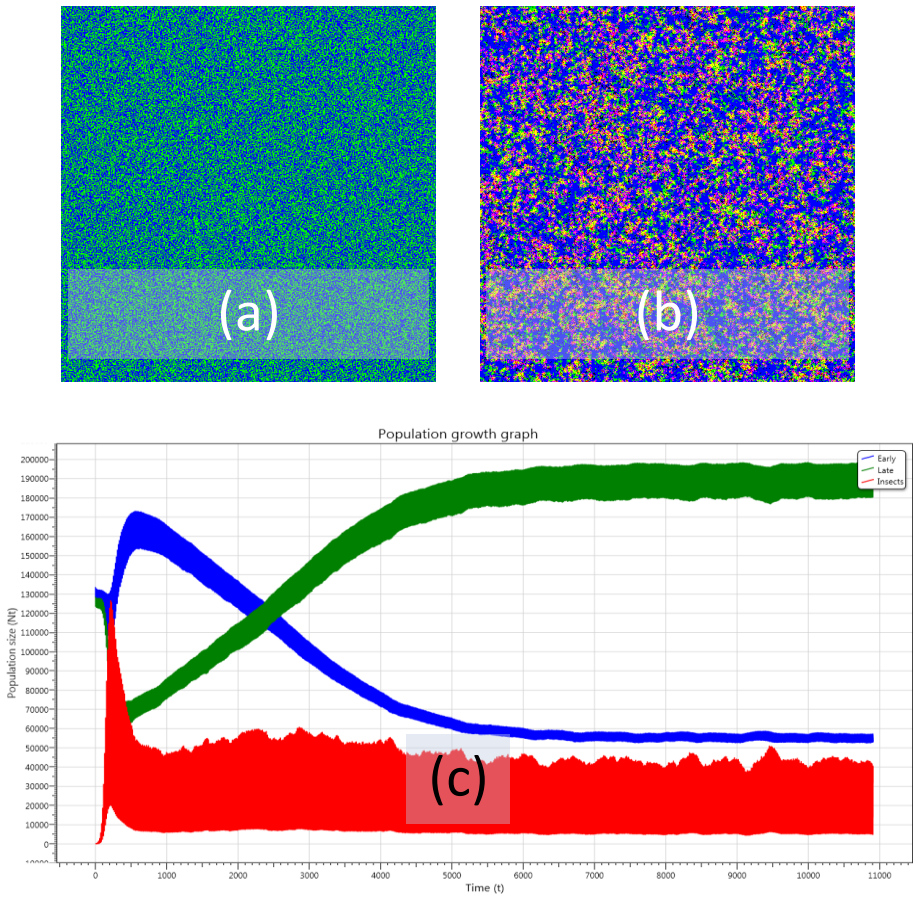


Figura 69. Consumo de recursos de los insectos. (a) Distribución inicial. (b) Al consumir recursos los insectos alteran su medio ambiente. (c) Dinámica del consumo.

4.3.3. Herbivory in minority species

Los parámetros de la simulación son los siguientes:

- Factor de mortalidad local = 1
- Porcentaje de mortalidad global = 0
- Influjo de insectos = 0 - 1
- Fronteras periódicas habilitadas
- Número de generaciones simuladas = 20000
- Densidad inicial de la especie temprana = 50%
- Densidad inicial de la especia tardía = 50%
- $\alpha_{e-l} = 0.250$
- $\alpha_{l-e} = 0.550$

El objetivo primordial del estudio de campo realizado por Futuyma y Wasserman, era dar un contraejemplo a un principio bien documentado denominado "Herbivory in majority species". Este principio establece que una especie numéricamente poco frecuente sufre una depredación menos intensa que una especie abundante. Sin embargo, los resultados muestran que en regiones dominadas por una cierta especie de roble, los predadores (insectos) afectan en mayor medida a la especie menos común.

En la Tabla 3 se observan con detalle estos resultados, en la columna "Área" se identifican con la letra S, regiones donde la especie temprana de robles es la dominante; de manera similar regiones donde la especie tardía es la dominante son denotadas con la letra W. En cada caso el nivel de daño promedio muestra que la especie minoría es más afectada por la acción de los predadores.

Tabla 3 Resultados obtenidos por Futuyma y Wasserman. Observe que en la gran mayoría de los casos la especie minoría sufre un mayor nivel de depredación (Adaptada de [28]).

Área	Sitio	Especie	Hojas usadas como muestra	Nivel de daño promedio	Alsophila por hoja	Larvas por hoja
S	1	Temprana	55	2.98	0.632	0.686
		Tardía	77	3.58	.217	0.236
	2	Temprana	57	3.72	0.493	0.514
		Tardía	---	5.00	---	
	3	Temprana	70	3.77	0.579	0.644
		Tardía	---	5.00	---	
	4	Temprana	71	3.92	0.614	0.643
		Tardía	---	5.00	---	
W	1	Temprana	141	2.94	0.100	0.119
		Tardía	95	2.61	0.161	0.256
	2	Temprana	50	1.72	0.193	0.219
		Tardía	87	1.81	0.052	0.075
	3	Temprana	58	2.66	0.198	0.220
		Tardía	104	1.48	0.037	0.050
	4	Temprana	54	2.67	0.097	0.166
		Tardía	69	1.29	0.023	0.123

Para observar este fenómeno en el modelo propuesto es necesario seguir la siguiente metodología:

a) Generar una distribución aleatoria del 50% de individuos de la especie temprana y 50% de individuos de la especie tardía.
b) Establecer condiciones de coexistencia. Los valores de α_{e-l} y α_{l-e} se han elegido de modo que ambas especies sean capaces de coexistir (ver sección 4.2.4), al mismo tiempo que una de ellas alcance un status de dominio (especie temprana).
c) Se simulan 5000 generaciones sin insectos, de modo que, cada especie alcance una capacidad de carga estable.
d) Se introducen insectos al 0.1% del total de árboles de la lattice.
e) Se simula 10000 generaciones adicionales.

En la Figura 70a Después de ser introducidos, la población de insectos tiene un rápido crecimiento, consecuencia de la abundancia de recursos (Figura 70b). En un ecosistema como este, donde hay pocos arboles de la especie tardía, solo un número reducido de insectos puede completar la migración con éxito; sin embargo, la mayor parte de los huevos puestos por los sobrevivientes residen en un árbol de la especie temprana (debido a su abundancia), lo que asegura que una gran proporción de la descendencia sobreviva para iniciar una nueva migración (Figura 70c).

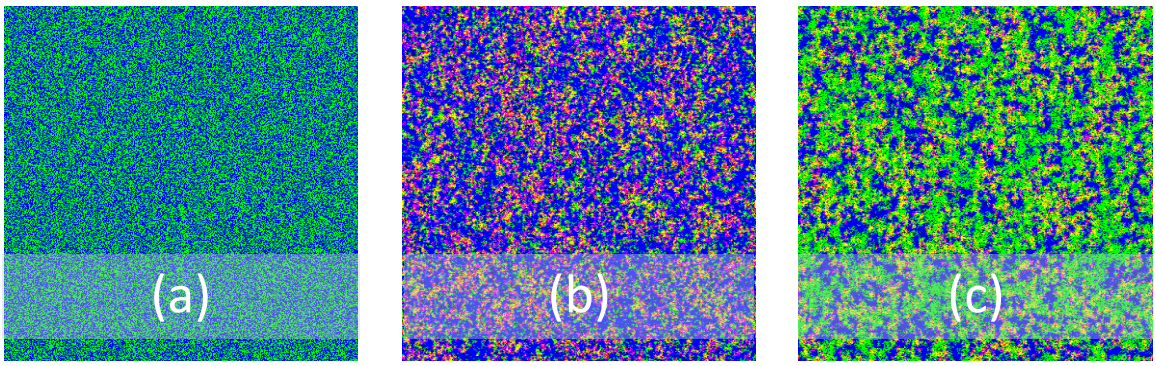


Figura 70. $(\boldsymbol{a})\ \boldsymbol{t = 0}\ (\boldsymbol{b})\ \boldsymbol{t = 5000}\ (\boldsymbol{c})\ \boldsymbol{t = 10000}$.

En la Figura 71 se ilustra la dinámica poblacional correspondiente a este fenómeno. Como se puede observar, la capacidad de carga de la especie menos abundante (especie tardía) disminuye debido a la actividad predadora de los insectos. Además, observe que la especie más abundante (especie temprana) aumenta su capacidad de carga al mismo tiempo que la de su competidora disminuye.

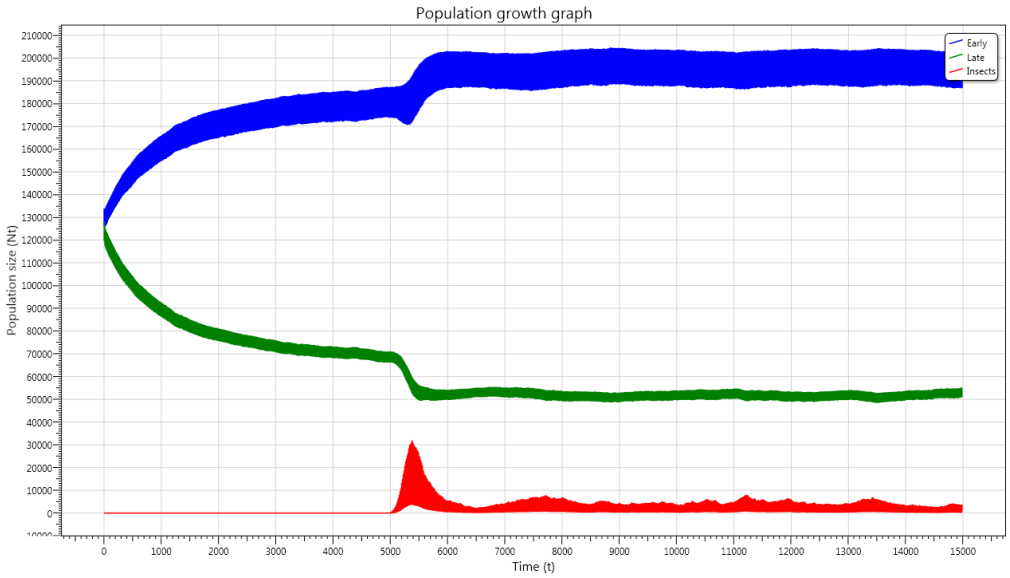


Figura 71. Dinámica del fenómeno "Herbivory in minority species".

El modelo propuesto es capaz de reproducir una dinámica compleja como la presente en el ecosistema descrito en [28]; sin embargo, lo hace no tratando de modelar el ecosistema en términos de los cambios de biomasa presentes en él, sino otorgando un papel primordial a las interacciones locales que los individuos tienen entre sí. El comportamiento global del ecosistema no es más que la "suma" de los efectos que cada individuo aporta.

4.4.Comparación con los modelos clásicos

4.4.1. Modelos clásicos de competencia intraespecífica

Si la especie tiene un comportamiento reproductivo discreto, esto es, se reproduce solo durante un cierto intervalo de tiempo, es posible modelar su comportamiento bajo los efectos de la competencia intraespecífica mediante una ecuación en diferencias con la siguiente forma:

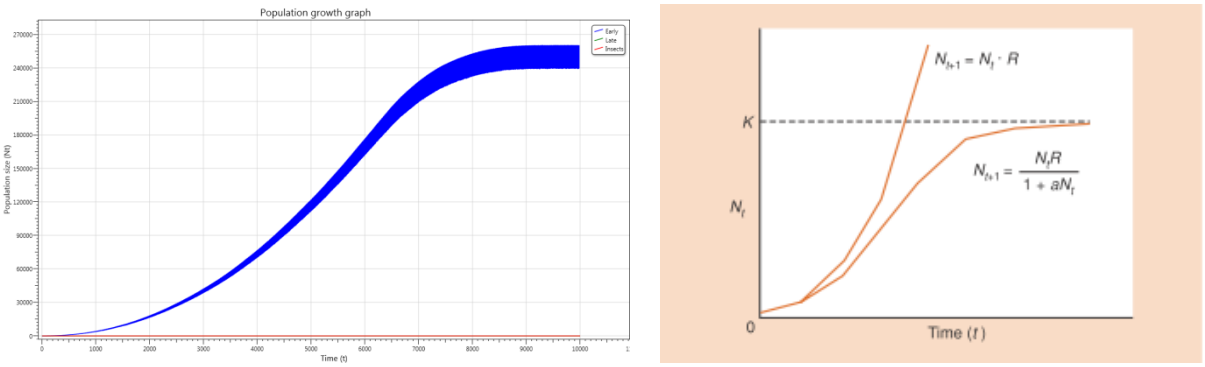
$$N_{t+1} = \frac{N_t R}{1 + \frac{(R-1)}{K} N_t} \qquad (4.5)$$

Dónde:

- N_t es el tamaño de la población en el tiempo t.
- R es la tasa neta de reproducción.
- K es la capacidad de carga de la población.

El tipo de competencia intraespecífica que la ecuación 4.5 es capaz de modelar es muy restringido, la población modelada alcanza siempre una capacidad de carga fija tras unas pocas iteraciones. En la Figura 72 se observa una comparación de las curvas de crecimiento poblacional para la ecuación 4.5 y para el modelo propuesto. Observe que a diferencia del modelo basado en ecuaciones en diferencias, la curva del modelo propuesto presenta fluctuaciones alrededor de un

punto de equilibrio, lo que se traduce en un grosor mayor de la curva en el punto donde se alcanza la capacidad de carga de la especie.



(a) (b)

Figura 72. Modelos de competencia intraespecífica. (a) Modelo propuesto. (b) Modelo de ecuaciones en diferencias [1].

Por supuesto, existen modelos de competencia intraespecífica más completos que el obtenido a partir de la ecuación 4.5. Estos modelos pretenden incorporar diversos aspectos omitidos por dicha ecuación, no obstante el modelo propuesto es capaz de reproducir una dinámica más completa a partir de reglas de evolución simples (ver sección 4.1).

Un problema fundamental con la ecuación 4.5 es que pareciera que el comportamiento de una población está determinado por R y K, entonces surge la pregunta ¿Cómo puede ser que el comportamiento de una población este determinado por una cantidad (K) que claramente es consecuencia de las condiciones imperantes en el ecosistema?

Si la especie tiene un comportamiento reproductivo continuo, su comportamiento puede ser modelado a través de la ecuación logística:

$$\frac{dn}{dt} = rN\left(\frac{K-N}{K}\right) \qquad (4.6)$$

La ecuación 4.6 es el equivalente continuo de la ecuación 4.5, en consecuencia comparte sus ventajas y desventajas. En la Figura 73 se observa la curva de crecimiento para un modelo continuo.

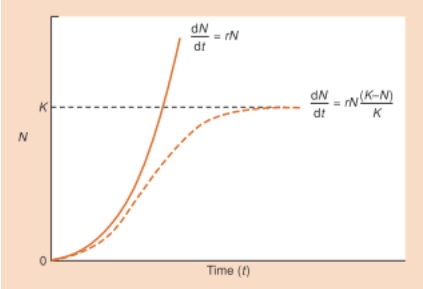


Figura 73. Dinámica poblacional para la ecuación logística [1].

4.4.2. Modelos clásicos presa – depredador

Considere el crecimiento de una población en la ausencia de consumidores, cuando esta población no está sujeta a los efectos de la competencia intraespecífica crece de manera exponencial de acuerdo a:

$$\frac{dN}{dt} = rN \tag{4.7}$$

Dónde:

- r es la tasa de reproducción.
- N es el tamaño de la población.

Sin embargo, en la presencia de depredadores, el tamaño de la población se ve afectado de manera proporcional al número de encuentros entre individuos de la especie presa e individuos de la especie predadora. Este tipo de encuentros se incrementa conforme el número de presas (N) y el número de predadores (P) crece, no obstante, un depredador no es un consumidor perfecto, y el número de presas consumidas depende de la eficiencia del mismo. Sea a la tasa a la que un depredador ataca a su presa, en consecuencia la tasa de consumo es aPN, y el crecimiento de una población está determinado por:

$$\frac{dN}{dt} = rN - aPN \tag{4.8}$$

La ecuación 4.8 es conocida como la ecuación Lotka – Volterra para la especie presa.

En ausencia de presas el número de depredadores en un ecosistema declina exponencialmente de acuerdo a:

$$\frac{dP}{dt} = -qP \tag{4.9}$$

Dónde:

- P es el tamaño de la población de predadores.
- q es la tasa de mortalidad de la especie predadora.

La reducción en la población de la especie predadora es contrarrestada por el nacimiento de nuevos individuos, la tasa de nacimientos depende de dos factores:

- La velocidad a la que se consumen presas (aPN).
- La eficiencia del depredador (f).

En consecuencia la tasa de nacimiento de los predadores es $faPN$, y la dinámica de la población de predadores está dada por:

$$\frac{dP}{dt} = faPN - qP \tag{4.10}$$

La ecuación 4.10 es llamada ecuación Lotka – Volterra de la especie predadora.

En la Figura 74 se muestra la dinámica presa – depredador en el modelo Lotka – Volterra. El tamaño de la población de predadores se incrementa cuando existe un número abundante de presas, cuando esto ocurre el consumo de presas se incrementa y el tamaño de la población disminuye. Esta disminución en el alimento de los predadores tiene como consecuencia una disminución en el tamaño de su población, lo que provoca que el consumo de presas disminuya, y en consecuencia la población de presas puede crecer nuevamente. Una de las desventajas principales del modelo Lotka – Volterra es que en la ausencia de un depredador, la especie presa crece indefinidamente.

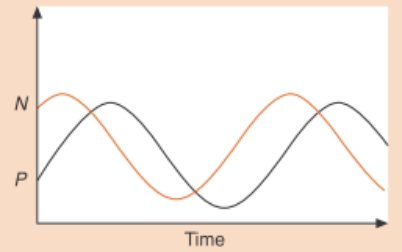


Figura 74. Dinámica presa – depredador en el modelo Lotka – Volterra [1].

En la Figura 75 se observa una curva de crecimiento típica para el modelo propuesto. Al igual que en el modelo Lotka – Volterra, podemos observar oscilaciones acopladas en las curvas de crecimiento para las especies presa (azul = especie temprana, verde = especie tardía) y la especie predadora (rojo).

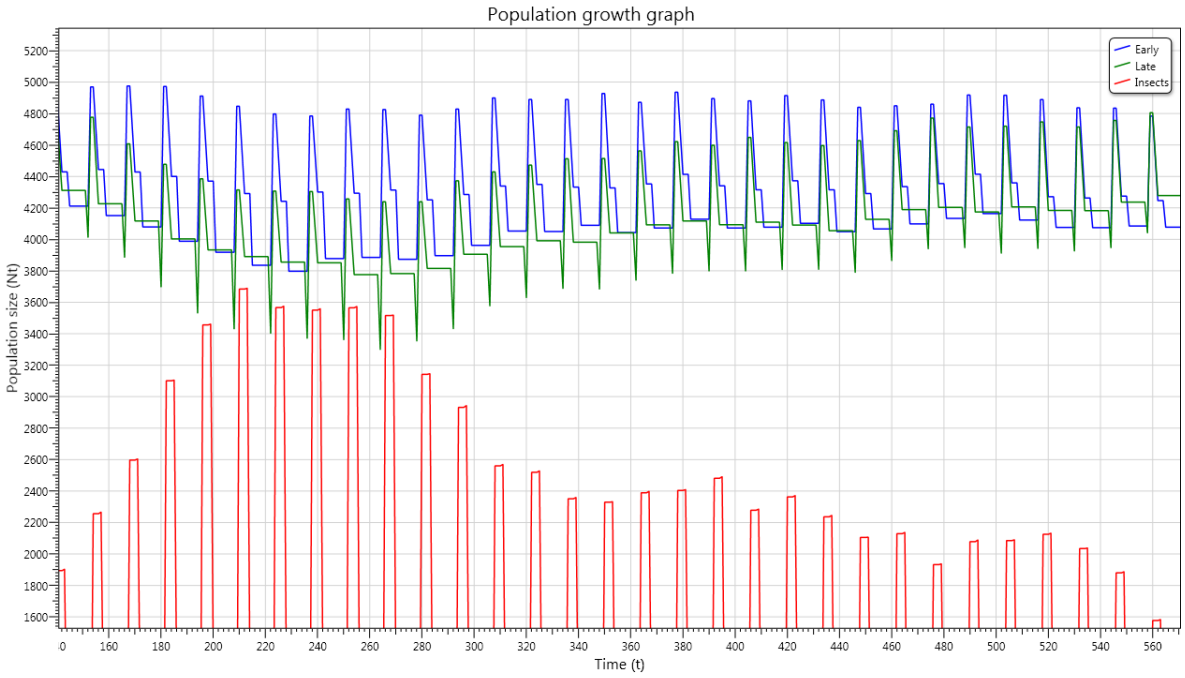


Figura 75. Dinámica de la depredación en el modelo propuesto.

Cuando existe un número abundante de individuos de las especies presa, la población de insectos puede crecer rápidamente, esta abundancia de insectos provoca un mayor consumo, y en consecuencia las poblaciones de presas disminuyen en tamaño. La reducción en el número presas provoca una reducción en el número de predadores, lo que permite el crecimiento de las poblaciones presa y el ciclo se repite. Sin embargo, como se mostró en la sección 4.3, en la depredación secuencial no basta que exista un número abundante de presas, ya que si la distribución de los individuos de estas especies no es la adecuada la población de predadores (insectos) se extinguirá. En conclusión, en un ecosistema bajo depredación secuencial no basta considerar los cambios de biomasa para modelar la dinámica de las interacciones entre presas y depredadores. En el modelo propuesto las interacciones locales entre presas y depredadores, dependiendo de los niveles de competencia establecidos, dan lugar a las distribuciones espaciales de las especies presa que determinan la supervivencia o extinción de la especie predadora.

Capítulo V. Modelado de la herramienta de software

La necesidad de modelar la interfaz gráfica de cualquier software surge naturalmente del desarrollo del mismo. Aún para escenarios muy simples, el modelado de la interfaz gráfica de usuario es esencial. A continuación se incluye la documentación del modelado de la interfaz gráfica de la herramienta de software. La metodología de modelado utilizada es la descrita en [34] y [35].

5.1.Análisis de requerimientos

5.1.1. Propósito del sistema

La herramienta a desarrollar debe proveer la simulación de un modelo ecológico basado en las ideas de Futuyma y Wasserman. La infraestructura de la herramienta es tal, que los conceptos matemáticos y computacionales subyacentes en la misma no representan un obstáculo para su uso.

La información generada y presentada por la aplicación deber ser tal, que el usuario pueda hacer uso de ella fácilmente, además, esta información debe poder ser accedida en un tiempo de cómputo razonable, de modo que la espera de la misma no constituya una limitante para las actividades del usuario.

5.2.Sistema propuesto

5.2.1. Requerimientos funcionales

Los requerimientos funcionales de la herramienta de software son los siguientes:

- La interfaz debe presentar en todo momento la evolución del autómata en la lattice.
- El tamaño máximo de la lattice debe ser 512 * 512 células.
- La interfaz gráfica debe proveer controles que permitan al usuario alterar los parámetros más significativos del modelo, de modo que, se puedan simular diversos escenarios.
- La aplicación debe incluir gráficas de dispersión donde se observe la variación en el tiempo de la densidad de población de robles blancos, robles escarlatas y larvas.
- La aplicación debe proveer de manera automática una dispersión inicial aleatoria de las tres especies en competencia de manera automática. De manera opcional el usuario podrá alterar esta dispersión a través del mouse y controles adicionales que la aplicación deberá proveer.
- La aplicación deberá proveer un mecanismo que permita obtener capturas gráficas de la aplicación para poder incluirlas en otros documentos. El formato de tales capturas deberá ser PNG (Portable Networks Graphics).
- Además del modelo de depredación secuencial, la aplicación deberá simular los siguientes modelos: autómatas celulares elementales, “Life”, modelo de difusión “HPP” y “Wire World”. La simulación de estos modelos ayudará al usuario a comprender la dinámica de los ACs.

5.2.2. Requerimientos no funcionales

a) Interfaz de Usuario.

- La representación gráfica de la evolución del autómata celular es presentada en una porción de la pantalla de 512×512 pixeles, facilitando de ese modo su interpretación.
- El tamaño de la lattice en la aplicación debe ser elegido de modo que el rendimiento no decaiga hasta el punto de volver inútil la aplicación.
- La interfaz gráfica de la aplicación será desarrollada utilizando el API WPF (Windows Presentation Foundation) de Microsoft.

b) Consideraciones de Hardware.

- Windows Vista o Windows 7.
- Microsoft .Net Framework 3.5.
- Procesador: Procesador X86 a 1 GHz.
- 256 MB de memoria RAM para la plataforma .Net 3.5.
- Tarjeta gráfica con soporte para Direct3D 9 y WDDM (Windows Display Driver Model)

c) Cuestiones de Calidad.

Cuantificar las características que definen un producto de software con el objetivo de determinar si se trata o no de un producto de calidad es una tarea complicada. El problema es que la mayoría de estas características se definen en una forma cualitativa, lo que dificulta su medición, ya que se requiere establecer métricas que permitan evaluar cuantitativamente cada característica dependiendo del tipo de software que se pretende calificar [36].

La norma ISO – 9126 establece un estándar internacional para la evaluación de productos de software. Este estándar establece que cualquier componente de la calidad del software puede ser descrito en términos de seis características básicas, al responder a una serie de preguntas que atienden cada una de estas características (véase la Tabla 4).

Tabla 4. Características de la norma ISO – 9126 y aspectos que atienden cada una.

Características	Pregunta Central
Funcionalidad.	¿Las funciones y propiedades satisfacen las necesidades explicitas e implícitas; esto es, el qué?
Confiabilidad.	¿Puede mantener el nivel de rendimiento, bajo ciertas condiciones y por cierto tiempo?
Usabilidad.	¿El software es fácil de usar y de aprender?
Eficiencia.	¿Es rápido y minimalista en cuanto al uso de recursos?
Mantenibilidad.	¿Es fácil de modificar y verificar?
Portabilidad.	¿Es fácil de transferir de un ambiente a otro?

A continuación se incluye un breve análisis de estas características y los aspectos más importantes de cada una de ellas en el desarrollo de la herramienta de software.

- Funcionalidad. La herramienta de software ha sido desarrollada en base a principios matemáticos bien establecidos, de acuerdo a esto, cumple con los objetivos establecidos para la misma. Reproduce de manera fiel los resultados experimentales, y proporciona al usuario un entendimiento más profundo acerca de la modelación con autómatas celulares.
- Confiabilidad. El rendimiento de la herramienta de software depende de varios factores tanto internos como externos. La evaluación de las miles de células que componen la lattice es tarea del procesador (CPU), sin embargo, la representación gráfica de las mismas es tarea de la Unidad de Procesamiento Gráfico. Para mantener un rendimiento constante es necesario mantener el intercambio de información entre estos dos componentes al mínimo. Solamente cuando este problema fundamental haya sido abordado durante la implementación del modelo, estaremos en condiciones de evaluar la confiabilidad de la herramienta de software.
- Usabilidad. Dado que se pretende que el software diseñado en esta tesis constituya una herramienta de apoyo para estudiantes, la facilidad de uso es un objetivo primordial y un parámetro de desarrollo que se seguirá. Sin embargo, solo cuando el desarrollo de la herramienta haya alcanzado cierto grado de madurez podremos evaluar debidamente esta característica.
- Mantenibilidad. La metodología de modelado utilizada para el desarrollo de la herramienta provee los modelos necesarios para analizar y modificar de manera eficiente al software. Sin embargo, carece de un análisis de riesgos adecuado, el cual debería ser incorporado a la metodología con el fin de alcanzar una buena gestión de mantenimiento.
- Eficiencia. La eficiencia del software estará íntimamente ligada a la complejidad computacional de los algoritmos utilizados para implementar el modelo del autómata celular, tomando en cuenta el problema mencionado en el análisis de confiabilidad. El uso de recursos solo podrá ser cuantificado cuando el modelo sea implementado y probado adecuadamente.
- Portabilidad. Dado que el software utiliza como plataforma de desarrollo Microsoft .Net, en principio sólo es posible su ejecución en ambientes Windows. Sin embargo existen implementaciones de los estándares que componen .Net en Linux y Mac, siendo el proyecto Mono el más completo.

5.2.3. Pseudo - requerimientos

- La plataforma de desarrollo será Microsoft .Net y el lenguaje de programación será C#.

5.3.Modelos del sistema

5.3.1. Modelo de casos de uso

El comportamiento del sistema bajo desarrollo (i. e., la funcionalidad que debe ser provista por el sistema) está documentado en un modelo de casos de uso que ilustra las funcionalidades del sistema (casos de uso), el entorno con el que interactúa (actores) y las relaciones entre los casos de uso y los actores (ilustradas por el diagrama de casos de uso) [37].

Los casos de uso que describen la funcionalidad de la herramienta son los siguientes:

- **Iniciar Nueva Simulación**. A través de este caso de uso el usuario inicia una nueva simulación.
- **Cargar Configuración de la Simulación**. A través de este caso de uso el usuario puede cargar la configuración de una nueva simulación almacenada en disco duro.
- **Modificar Parámetros de la Simulación**. A través de este caso de uso el usuario puede alterar diversos parámetros de la simulación.
- **Guardar Configuración de la Simulación**. A través de este caso de uso el usuario puede guardar la configuración actual de la simulación en disco duro.

La Tabla 5 muestra la descripción detallada del caso de uso Iniciar Nueva Simulación.

Tabla 5. Documentación del caso de uso Iniciar Nueva Simulación.

1.	Nombre del Caso de Uso	Iniciar Nueva Simulación
	1.1. Actores Participantes	Usuario
	1.2. Breve Descripción	A través de este caso de uso el usuario inicia una nueva simulación.
2.	**Flujo de Eventos**	
	2.1. Flujo Básico	Este caso de uso inicia cuando el usuario selecciona el modelo que desea simular, así como los parámetros adecuados para la misma. Una vez que los parámetros han sido establecidos, el usuario puede altera la distribución inicial de la lattice e iniciar la simulación a través de los controles establecidos para ese fin.
	2.2. Flujos Alternos	
	2.2.1.	No hay flujos alternos para este caso de uso.
3.	**Requerimiento Especiales**	
	3.1. Requerimiento Especial 1	Los parámetros del modelo han sido establecidos.
4.	**Precondiciones**	
	4.1. Precondición 1	No hay precondiciones para este caso de uso
5.	**Post Condiciones**	
	5.1. Post Condición 1	No hay post condiciones

Las relaciones entre el usuario y los casos de uso de la herramienta pueden observarse en el diagrama de casos de uso de la Figura 76.

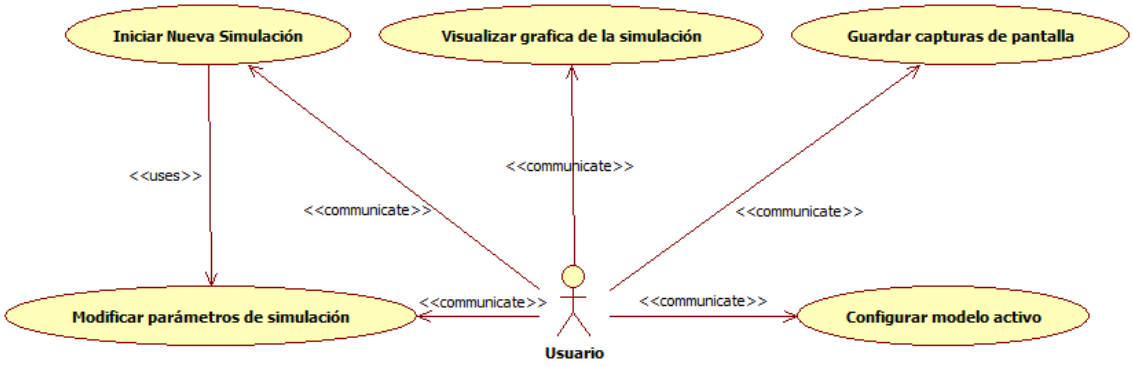


Figura 76. Diagrama de casos de uso para la herramienta de software.

5.3.2. Modelo de dominio

A partir de los casos de uso y la especificación del sistema, se obtienen el diseño del modelo de dominio, cuya representación está dada por el diagrama de clases UML de la Figura 77. Este diagrama de clases está compuesto de clases $\ll actor \gg$ y $\ll entity \gg$ que corresponden al usuario, el Modelo de la herramienta de software y el autómata celular.

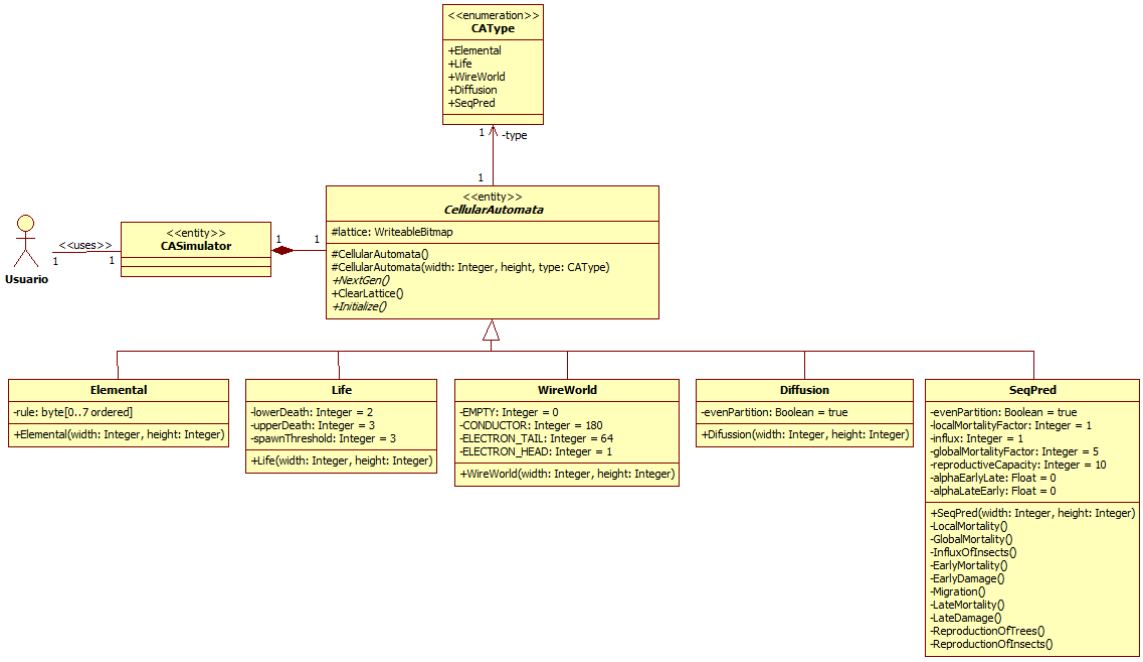


Figura 77 Diagrama de clases UML del modelo de dominio.

5.3.3. Modelo de presentación abstracto

Tras la especificación del modelo de dominio, aun no es necesaria una especificación detallada del modelo de la interfaz gráfica de usuario (GUI por sus siglas en ingles), solo es necesario conocer el tipo de componentes que requiere esta interfaz, cuántos son y cómo podrían

ser agrupados. Es decir, las especificaciones de este modelo son realizadas con un alto nivel de abstracción.

Para ejemplificar el uso del modelo de presentación abstracto, considere el diagrama de secuencia de la Figura 78. El actor $Usuario$ inicia la interacción al enviar el mensaje $ejecutar$ a la aplicación, este mensaje es el resultado de la interacción necesaria entre el usuario y el sistema operativo para iniciar la ejecución de la aplicación (e. g. un doble clic en un icono). En respuesta a este mensaje la aplicación crea una instancia de la clase $CellularAutomata$ (esta instancia puede ser de cualquiera de los tipos contenidos en la enumeración $CAType$). La creación de un objeto es modelada a través del envió de un mensaje $\ll create \gg$ al nuevo objeto [34]. Después de que esta instancia ha sido creada, es necesario crear la UI necesaria para que el usuario pueda interactuar con el modelo creado, este proceso es modelado a través de la llamada a $CreateUIForCA()$. Hecho esto, el usuario puede utilizar la GUI para establecer los parámetros de la simulación. Esto es modelado a través del mensaje $establecer\ parámetros$, en primera instancia este mensaje es enviado a la aplicación, la cual a su vez tiene la tarea de enviarlo a la instancia de la clase $CellularAutomata$. Finalmente el usuario envía un mensaje a la aplicación indicando su deseo de iniciar la simulación, este mensaje se traduce en una llamada a la operación $NextGen()$ de la instancia de $CellularAutomata$.

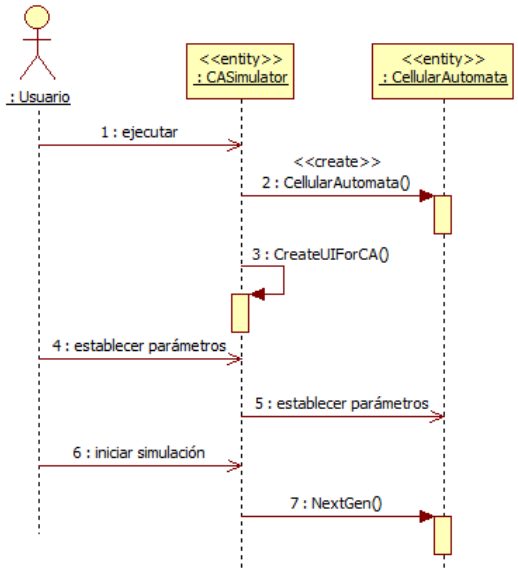


Figura 78. Diagrama de secuencia para el caso de usos "Iniciar Simulación".

En la Figura 79 se muestra el diagrama de clases que es utilizado como base para describir una interfaz de usuario. Este diagrama es la especificación del modelo de presentación abstracto, en este sentido todas las clases contenidas en este diagrama son calificadas utilizando el estereotipo $\ll apm \gg$ (abstract presentation model). En [35] se discute en detalle la categorización de componentes abstractos, para describir la interface de la aplicación $CASimulator$, solo se consideran las siguientes categorías:

- $AbstractContainer$. Se trata de un objeto capaz de agrupar varios $AbstractComponent$, así como otros contenedores $AbstractContainer$. La

especialización $AbstractForm$ representa el contenedor de más alto nivel en el modelo de presentación abstracto.

- $AbstractComponent$. Representa un componente genérico que a su vez se divide en las siguientes categorías
    a) $StaticDysplay$. En esta categoría se encuentran componentes que solo proveen algún tipo de información visual, tales como etiquetas o imágenes estáticas.
    b) $ActionInvoker$. Son componentes que provocan eventos que pueden ser descritos como operaciones del sistema, e. g. botones.
    c) $InteractionControl$. En esta categoría se encuentran componentes que generan eventos, que modelan opciones para el usuario y que constituyen la navegabilidad de la interface, e. g. menús.

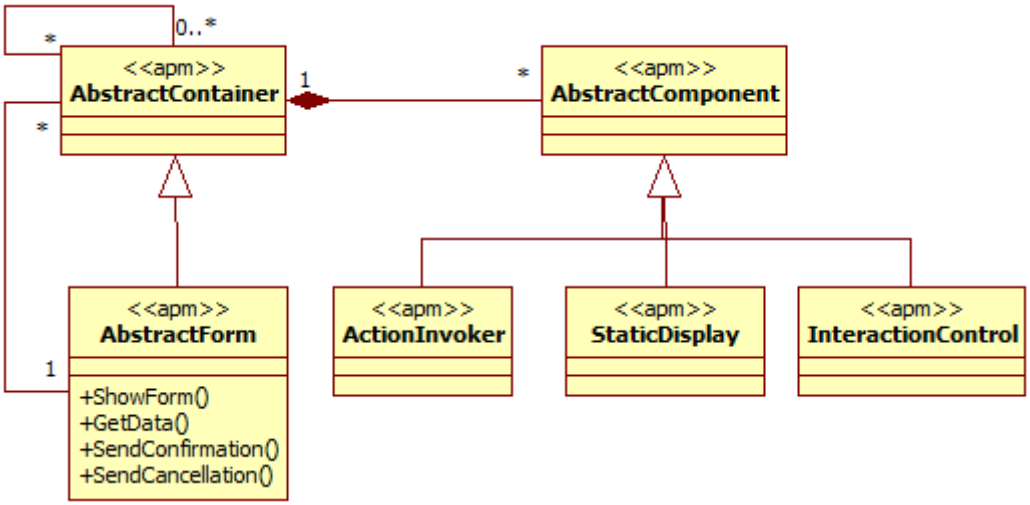


Figura 79. Diagrama de clases UML del modelo de presentación abstracto.

Cuatro operaciones son definidas en la clase $AbstractForm$, estas operaciones son suficientes para proveer la funcionalidad básica que los objetos $\ll boundary \gg$ deben implementar. Estas operaciones son las siguientes:

- $ShowForm()$. Es un mensaje auto – delegado, que tiene como función dibujar la "forma" en el dispositivo de salida.
- $GetData()$. Recolecta la información provista por el usuario después de su interacción con la aplicación. Además realiza los procedimientos adecuados para transformar esta información en parámetros que sean utilizables por las operaciones del sistema.
- $SendConfirmation()$ y $SendCancellation()$. Responsables de informar respectivamente que el usuario desea enviar información al sistema, o informar que el usuario desea terminar la interacción con un objeto $\ll boundary \gg$ sin enviar información alguna.

Para ejemplificar el uso del modelo de presentación abstracto, considere la ventana de diálogo de la Figura 80. El usuario interactúa con esta ventana para establecer el modelo de AC activo en la aplicación.

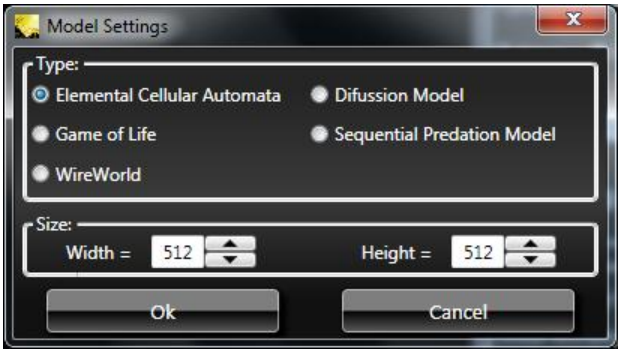


Figura 80. Ventana de diálogo "Model settings".

El diagrama de clases de la describe de manera conceptual la ventana de diálogo de la Figura 80. Los enlaces etiquetados con $compose$ son los que relacionan $AbstractComponents$ con $AbstractContainers$. Los enlaces entre dos instancias de $AbstractContainer$ son etiquetados con $integrate$.

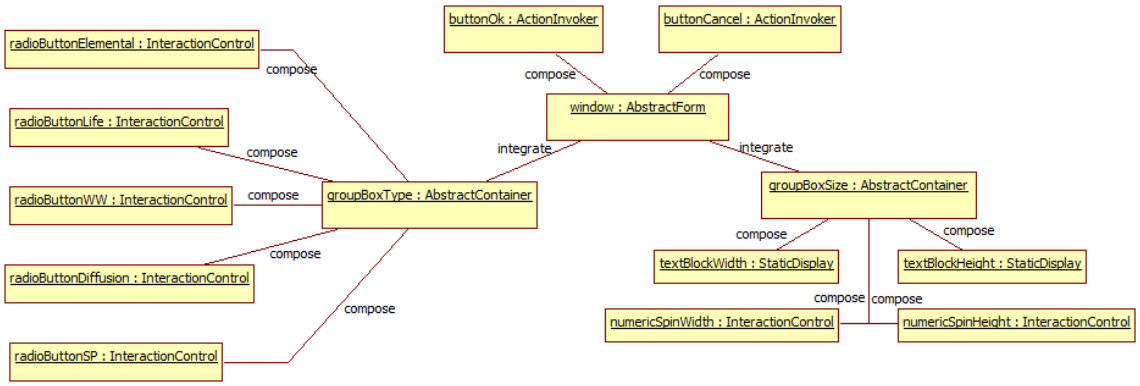


Figura 81. Modelo de presentación abstracto para la ventana de diálogo "Model settings".

5.3.4. Modelo de interfaz de usuario (UI)

El modelo de interfaz de usuario está compuesto por clases $\ll boundary \gg$, cuya tarea principal es administrar la interacción entre el usuario y el sistema. En la Figura 82 se observa el diagrama de clases del modelo de interfaz de usuario, observe que la aplicación $CASimulator$ está compuesta por una ventana llamada $MainWindow$. Esta ventana es donde la mayor parte de la interacción entre el usuario y la aplicación toma lugar, esta clase además contiene dos campos para las ventanas $WindowModel$ y $WindowGraph$, a través de estas ventanas el usuario puede respectivamente cambiar el modelo de CA activo en la aplicación u observar la gráfica del modelo activo.

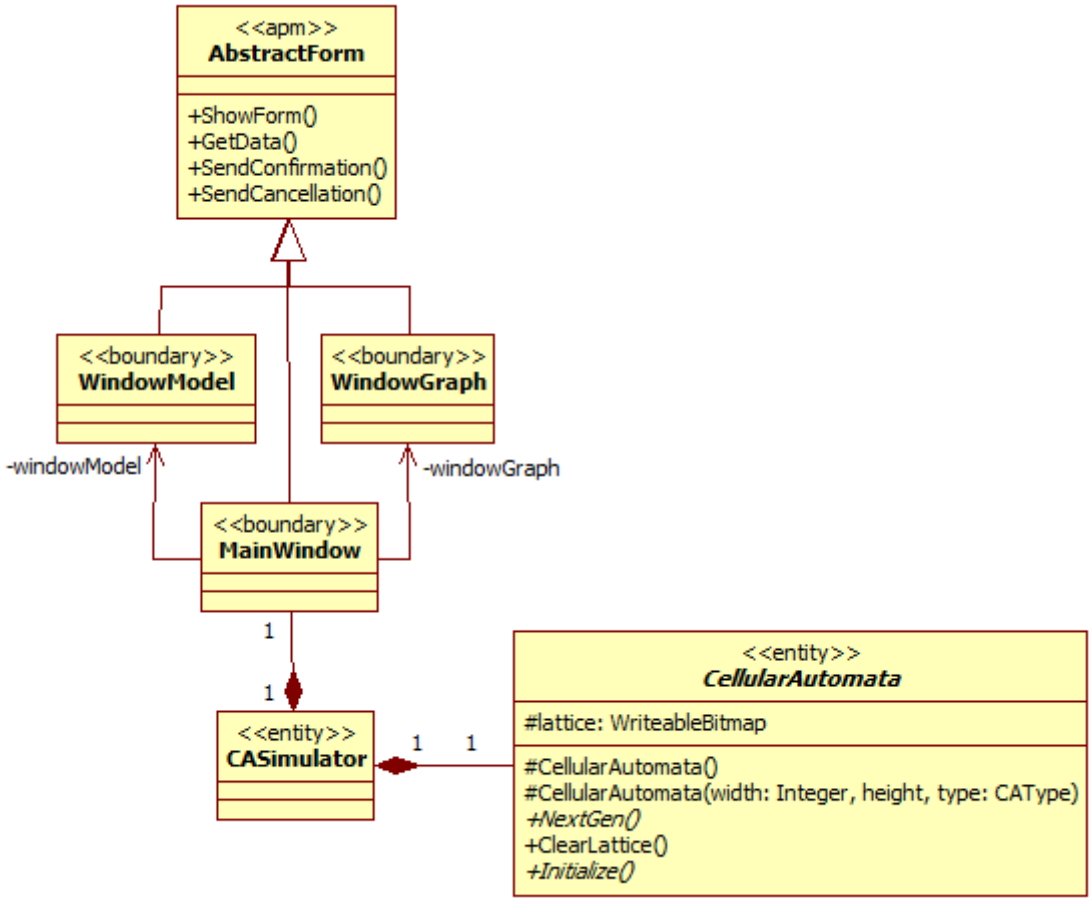


Figura 82. Diagrama de clases del modelo de interfaz de usuario.

5.3.5. Modelo de presentación concreto.

El diagrama de la Figura 79 es la especificación del patrón de diseño para el modelo de presentación concreto de la GUI. A partir de este modelo, se obtendrá un diagrama de clases que describe como los componentes de "entorno" pueden ser agrupados de acuerdo a los diagramas del modelo de presentación abstracto. Dentro de este contexto, un "entorno" representa las clases de un lenguaje de programación o los componentes de una plataforma de desarrollo de software como J2SE o Microsoft .Net. Debido a esto el modelo de presentación concreto es dependiente del entorno, y es descrito en términos de clases y componentes entorno calificados por el estereotipo $\ll cpm \gg$.

La Figura 83 muestra el diagrama de clases del modelo de presentación concreto, para su elaboración se han utilizado clases pertenecientes a WPF, observe el símbolo de colaboración del modelo de presentación abstracto. Para mantener la claridad, en la Figura 83 solo se han indicado las dependencias de algunas clases. Sin embargo, clases como $RadioButton$ deben incluir las dependencias pertinentes que indiquen su relación con el modelo de presentación abstracto, en el caso de $RadioButton$, la dependencia faltante es $InteractionControl$.

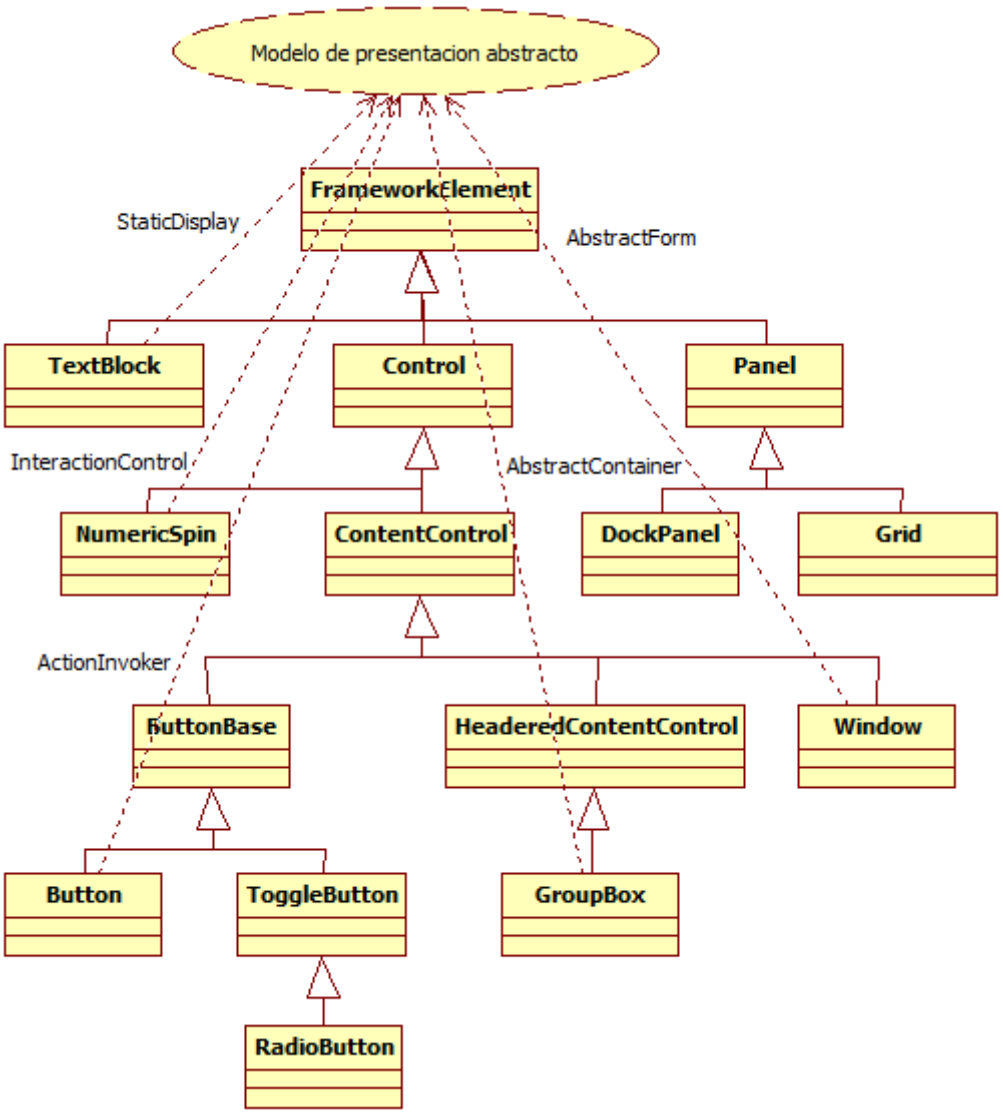


Figura 83. Diagrama de clases UML del Modelo de Presentación Concreto.

5.3.6. Modelo de entorno

En el Modelo de Entorno se incluyen aquellas clases utilizadas para la construcción del sistema de software. El entorno puede ser un lenguaje de programación orientado a objetos tal como C++ o Java, incluso puede tratarse de un conjunto de componentes y lenguajes orientados a objetos. El entorno se ocupa principalmente de la parte visual de la interfaz de usuario, pero podría ser responsable del comportamiento más importante de la aplicación, sobre todo cuando se consideran el uso de componentes complejos.

La herramienta modelada en la presente investigación, utiliza como plataforma de programación Microsoft .Net y como subsistema gráfico / API Windows Presentation Foundation (WPF). Dada la gran complejidad inherente a ambas entidades, en el modelo de entorno sólo se incluyen las clases más importantes utilizadas en el desarrollo de la herramienta. En concreto, se incluyen las clases descritas en la Tabla 6.

Tabla 6. Clases de Entorno utilizadas en el modelado de la aplicación.

Nombre de la clase	Espacio de Nombres	Descripción
$\boldsymbol{BitmapSource}$	System.Windows.Media.Imaging	Representa un conjunto inmutable de pixeles de un cierto tamaño y resolución.
$\boldsymbol{Color}$	System.Windows.Media	Describe un color en términos de los canales alfa, rojo, verde y azul.
$\boldsymbol{Decorator}$	System.Windows.Controls	Provee una clase base para aquellos elementos que aplican efectos sobre o alrededor de sus controles hijos.
$\boldsymbol{DispatcherTimer}$	System.Windows.Threading	Representa un temporizador integrado en la cola System.Windows.Threading.Dispatcher, el cual es procesado en un intervalo dado, con una prioridad especifica.
$\boldsymbol{Image}$	System.Windos.Controls	Representa un control capaz de desplegar una imagen.
$\boldsymbol{Matrix}$	System.Windows.Media	Representa una matriz de transformación afina utilizada para realizar transformaciones en un espacio bidimensional.
$\boldsymbol{ObservableCollection<T>}$	System.Collection.ObjectModel	Representa una colección de objetos dinámica, que provee notificaciones cuando se agregan, remueven o se actualizan los ítems contenidos.
$\boldsymbol{Point}$	System.Windows	Representa un par coordenado x, y en un espacio bidimensional.
$\boldsymbol{ViewBox}$	System.Windows.Controls	Define un contenedor "decorador" simple, que puede escalar su contenido para llenar el espacio disponible.
$\boldsymbol{WriteableBitmap}$	System.Windows.Media.Imaging	Provee una instancia $BitmapSource$ que puede ser escrita y actualizada. Esta clase es especialmente útil cuando se desea generar el contenido de una imagen de manera algorítmica (e. g. en un fractal).

5.3.7. Modelo de control

El modelo de control se utiliza para ejemplificar el uso de los Modelos de Presentación Abstracto y Concreto. En este modelo se incluyen los siguientes diagramas:

- Diagramas de Secuencia. En estos diagramas se muestran los objetos y clases involucrados en un caso de uso y la secuencia de mensajes necesarios que son intercambiados entre los objetos, para llevar a cabo la funcionalidad provista por el caso de usos, e. g. el diagrama de secuencia de la Figura 78.
- Diagramas de Actividades. Estos diagramas representan la parte dinámica de la herramienta. Muestran el flujo de una actividad a otra en el sistema, qué actividades pueden realizarse en paralelo, además de cualquier flujo alterno que exista dentro de la actividad.

La Figura 84 muestra el diagrama de actividades de la herramienta, en el mismo se observa el flujo de actividades de alto nivel que la componen.

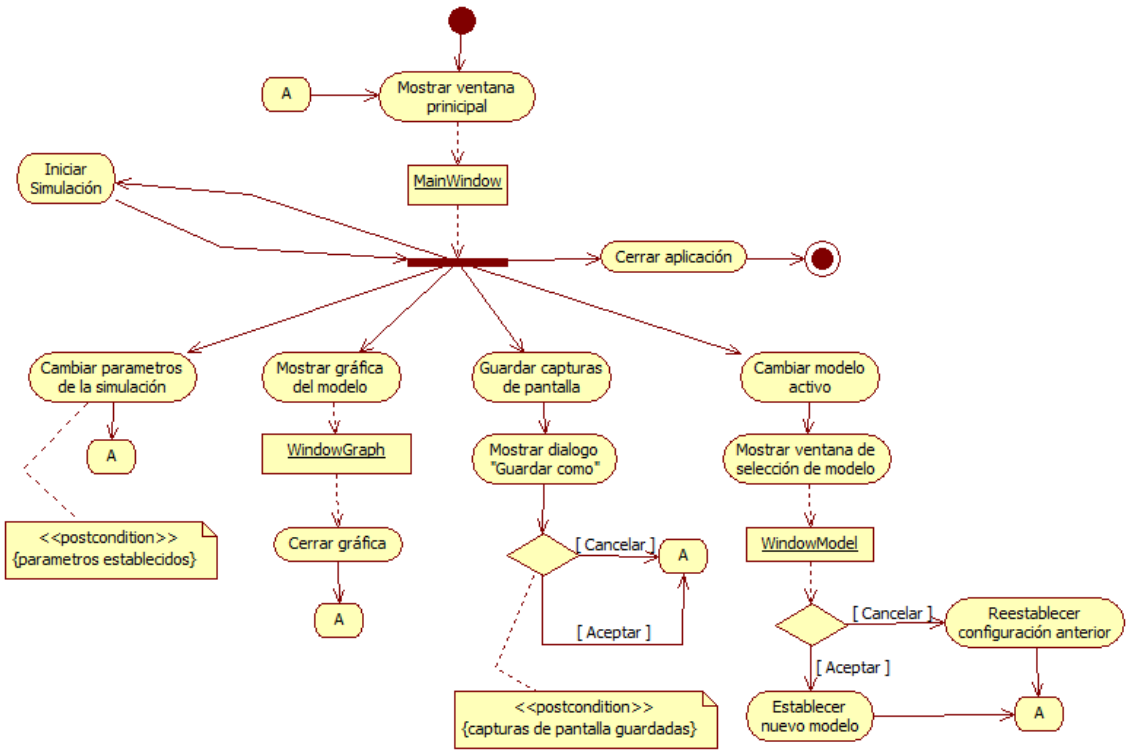


Figura 84. Diagrama de actividades de la aplicación.

5.3.8. Empaquetado de la aplicación

La Figura 85 muestra el diagrama de paquetes que provee una visión general del modelado de la aplicación. Además, en este diagrama es posible observar las dependencias entre los diferentes componentes del sistema.

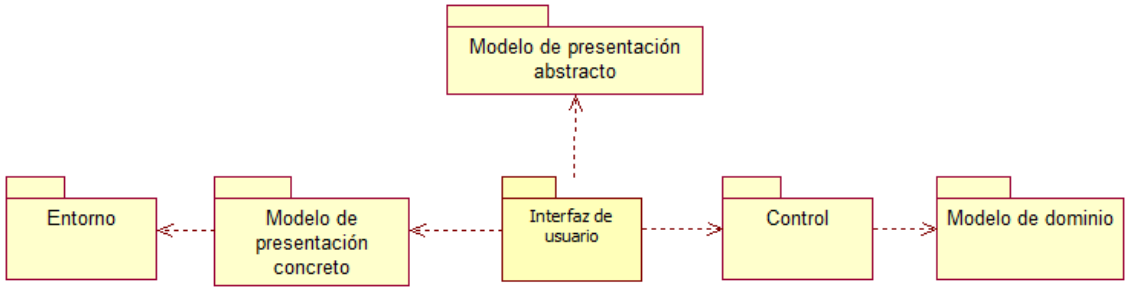


Figura 85. Diagrama de paquetes de la aplicación.

Capítulo VI. Conclusiones

Sin duda una de las cosas más impresionantes de los autómatas celulares es la riqueza y complejidad del comportamiento que exhiben, sin embargo, no menos hermoso e interesante es su capacidad para describir la naturaleza en términos que los modelos convencionales son incapaces de hacer, como Joel L. Schiff dijo *"Los autómatas celulares son una visión discreta del mundo"*.

A partir de la realización de este trabajo, se obtuvieron las siguientes conclusiones:

- Se realizó la investigación necesaria sobre autómatas celulares (clásicos y de bloque), redes de Petri y autómatas probabilísticos, para resolver el problema bajo estudio.
- Se modelaron las interacciones locales de los siguientes esquemas de cómputo a través de una red de Petri:
    a) Autómatas finitos probabilísticos.
    b) Autómatas celulares clásicos.
    c) Autómatas celulares de bloque.
- Se desarrolló un modelo de competencia intraespecífica y un modelo de competencia interespecífica que son coherentes con las observaciones empíricas.
- Se desarrolló un modelo de depredación secuencial que reproduce adecuadamente los resultados del estudio de campo de Futuyma y Wasserman.
- Se discutieron y analizaron los patrones espaciales generados por el modelo, además, dichos patrones han sido colocados en un contexto ecológico.
- Al considerar las interacciones entre individuos como parte fundamental de la dinámica de un ecosistema, el modelo propuesto se coloca en una perspectiva ecológica más realista.
- Mediante una rigurosa experimentación se han determinado los valores de los coeficientes de competencia, necesarios para reproducir el fenómeno de "Herbivory in minority species".
- Se ha desarrollado un sistema de software en la plataforma Microsoft .Net, que simula el modelo propuesto.

Capitulo VII. Trabajo Futuro

El trabajo de tesis presentado puede ser mejorado en varios aspectos, tanto teóricos, como técnicos. A continuación se presentan las mejoras propuestas.

7.1.Aspectos Teóricos

- El modelo de competencia intraespecífica propuesto exhibe equilibrio asintótico estable, sin embargo numerosos estudios en la teoría ecológica muestran poblaciones que además de exhibir este tipo de comportamiento, muestran una dinámica que puede ir desde un comportamiento monotónico hasta el caos. Es necesario desarrollar un modelo de competencia intraespecífica que sea capaz de reproducir este tipo de dinámicas, como Wolfram ha demostrado, este tipo de comportamiento puede ser simulado a través de ACs, lo que resulta alentador para el desarrollo de la teoría ecológica a través de modelos espaciales.

- A través del modelo propuesto es posible observar el comportamiento de dos especies bajo condiciones de competencia intraespecífica e interespecífica, así como de depredación. Además, se han analizado las condiciones que permiten la coexistencia o exclusión de las especies presentes en el ecosistema. Sin embargo, dado que el estudio de campo de Futuyma y Wasserman carece de datos estadísticos que permitan determinar cuantitativamente los niveles de competencia intraespecífica e interespecífica presentes en ese ecosistema, resulta imposible analizar la exactitud del modelo propuesto a un nivel cuantitativo. Es necesario pues contar con esta información, si este tipo de análisis ha de ser realizado, en palabras de Michael Begon *"La Ecología debe aspirar a ser una ciencia predictiva".*

- Para poder analizar el comportamiento del sistema, es necesario desarrollar una metodología que permita establecer métricas para medir los niveles de competencia intraespecífica e interespecífica.

- El algoritmo de migración utilizado se basa en el modelo HPP, en consecuencia, este algoritmo solo se ocupa del movimiento de las partículas, sin tomar en cuenta la razón subyacente para el movimiento de las mismas. Por ejemplo los insectos migran debido a la necesidad de acceder a nuevos recursos para poder sobrevivir, es lógico pensar que una vez que los encuentran, desaparece la necesidad de migrar. Utilizar un algoritmo de difusión para simular este comportamiento, resulta en una "migración tonta", debido a que los insectos pueden hallarse ya en un roble blanco, sin embargo, debido al algoritmo de difusión seguirán migrando "ciegamente", hasta que la difusión se detenga. Desarrollar un algoritmo inteligente de migración representaría un avance significativo, si se desea simular una migración más cercana a la realidad.

- El comportamiento propuesto para la especie predadora (insectos), fue modelado con el propósito de simular el fenómeno de "herbivory in minority species", en alcanzar este objetivo, el modelo propuesto ha sido exitoso. Sin embargo, con el objetivo de desarrollar una verdadera teoría ecológica basada en ACs, es necesario modelar el comportamiento de los predadores en un nivel más general, de modo que, del mismo modo que se hizo para las especies presa, sea posible reproducir las dinámicas más representativas de la depredación.

7.2.Aspectos Técnicos

- Con la disponibilidad de la plataforma .Net 4.0 existen nuevas tecnologías que posibilitan la programación concurrente en una manera más eficiente y segura. A través de un modelo de programación basado en "Tasks" (tareas) es posible dividir una carga de trabajo en unidades de ejecución concurrentes, con esta nueva API es posible solucionar los problemas clásicos de la escalabilidad, el desbalance de carga y la sobre explotación de hilos. Utilizar el nuevo API de la plataforma .Net 4.0 en la implementación del modelo propuesto significaría un incremento notable en el desempeño de su simulación.

- Aunque se pueden seguir varias estrategias de paralelización gracias a la disponibilidad de procesadores de varios núcleos, el número masivo de células en una lattice típica sobrepasa por mucho la capacidad de estos procesadores. Sin embargo, gracias al reciente auge en el desarrollo de los procesadores gráficos (Graphics Processing Unit), se tiene una plataforma adecuada para la simulación de los modelos de CAs.

- El uso de un GPU abre la posibilidad de una implementación tridimensional del modelo propuesto, esto facilitaría el análisis de los patrones espaciales generados por el modelo, y abre la posibilidad de observar patrones que no se dan en una implementación bidimensional.

- El modelo propuesto incorpora varias reglas probabilísticas, en su implementación se utilizó el PRNG (Pseudo Random Number Generator) contenido en el espacio de nombres System de la plataforma .Net. Este PRNG se basa en las ideas de Donald E. Knuth publicadas en [38]. Aunque en general este PRNG se considera "eficiente", en la actualidad se buscan algoritmos alternativos con periodos más largos y distribuciones probabilísticas uniformes, un ejemplo de un algoritmo que tiene un periodo de $2^{19937} - 1$ es descrito en [39]. Este algoritmo constituye una buena alternativa, de modo que se obtengan resultados confiables en términos de una buena distribución probabilística.

Bibliografía

1. **Begon, Michael, Townsend, Colin R. y Harper, John L.** *Ecology: From Individuals to Ecosystems.* Oxford, UK : Blackwell Publising, 2006. 1-405-1117-8.

2. **Hogeweg, Paulien (c).** *Cellular Automata as a Paradigm for Ecological Modeling.* s.l. : Applied Mathematics and Computation. Elsevier, 1988. págs. 81-100. Vol. 27.

3. **Chopard, Bastien y Droz, Michel.** *Cellular Automata Modelling of Physical Systems.* Cambridge, U. K. : Cambridge University Press, 1998. 0-521-46168-5.

4. **Schiff, Joel L.** *Cellular Automata: A Discrete View of the World.* New Jersey : Wiley, 2008. 978-0-470-16789-0.

5. **Zamora, R. R.** *Modelación de Flujo de Tránsito de Autos Utilizando Autómatas Celulares.* s.l. : CINVESTAV - IPN, 2002.

6. **Novoa Cataño, Javier.** *Analisis de la Probabilidad de Accidentes Vehiculares en un Modelo Propuesto de Flujo de Tráfico Vehicular Multi - Carril.* México, D. F. : CIC - IPN, 2007.

7. **Brauer, Wilfried y Wolfgang, Reisig.** *Carl Adam Petri und die, Petrinetze.* s.l. : Informatik Spectrum. Springer Berlin / Heidelberg, Octubre 2006. págs. 369-381. Vol. 29. 0170-6012.

8. **Peterson, James L.** *Petri Nets.* New York : ACM Computer Surveys. ACM, 1977. págs. 223-252. Vol. 9. 0360-0300.

9. **Tadao, Murata.** *Petri Nets: Properties, Analysis and Applications.* Chicago Illinois : Proceedings of the IEEE, 1989. págs. 541-581. Vol. 77. 8926700.

10. **Zhenzhi, Lin, y otros.** *A Survey on the Applications of Petri Nets Theory in Power Systems.* China : IEEE, 2006.

11. **Holloway, L. E., Krogh, B. H. y Giua, A.** *A survey of Petri Net Methods for Controlled Discrete Event Systems.* Boston : Discrete Event Dynamic Systems. Springer Netherlands, 1997. págs. 151-190. Vol. 7. 0924-6703.

12. **Pinney, J. W., Westhead, D. R. y McConkey, G. A.** *Petri Net representations in system biology.* U. K. : Biochemical Society Transactions, 2003. págs. 1513-1515. Vol. 31.

13. **Bobbio, Andrea.** *System Modelling with Petri Nets.* Kanpur, India : System Reliability Assesment: Proceedings of the ISPRA. Kluwer Academic Publishers, 1990. págs. 103-144. 0-7923-0837-9.

14. **Zurawski, Richard y Zhou, MengChu.** *Petri Nets and Industrial Applications: A Tutorial.* s.l. : IEEE Transactions on Industrial Electronics, 1994. págs. 567-583. Vol. 41. No. 6.

15. **Michaels, John G., Rosen, Kenneth H. y McGuigan, Robert A.** *Applications of Discrete Mathematics.* New York, USA : McGraw Hill, 2007.

16. **Bause, Falko y Pieter, Kritzinger.** *Stochastic Petri Nets: An Introduction to the Theory.* 2. Dortmund, Germany : Informatik IV. Vieweg Verlag, 2002. 3-528-15535-3.

17. **Cassandras, Christos G. y Lafortune, Stéphane.** *Introduction to Discrete Event Systems.* Segunda. s.l. : Springer, 2008. 978-0-387-33332-8.

18. **Sokolova, Ana y de Vink, Erik P.** *Probabilistic Automata: System Types, Parallel Composition and Comparison.* s.l. : Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004. págs. 377-385. Vol. 2925/2004. 0302-9743.

19. **Volterra, Vito.** *Fluctuations in the abundance of species considered mathematically.* s.l. : Nature, 1926. págs. 558-560. Vol. 118.

20. **Lotka, Alfred.** *The growth of mixed species: Two species competing for a common food supply.* s.l. : Journal of the Washington Academy of Sciences, 1932. págs. 461-469. Vol. 22.

21. **Wolfram, Stephen.** *A New Kind of Science.* s.l. : Wolfram Media Inc, 2002.

22. **Molofsky, J.** *Population Dynamics and pattern formation in theoretical populations.* s.l. : Ecology 75: 31-39, 1994.

23. **Wootton, JT.** *Local Interactions predict large - scale pattern in empiracally derived cellular automata.* s.l. : Nature 413: 841-844, 2001.

24. **Hogeweg, Paulien (a).** *From population dynamics to ecoinformatics: Ecosystems as multilevel information processing systems.* Utrecht University, Netherlands : Ecological Informatics. Elsevier, 2007. págs. 103-111. Vol. 2. 1574-9541.

25. **Hogeweg, Paulien (b).** *Multilevel Process in evolution and development: Computational models and biological insights.* s.l. : Lecture Notes in Physics. Multilevel Process in evolution and development: Computational Models and biological insights. Springer Berlin / Heidelberg, 2002. págs. 217-239. Vol. 585. 978-3-540-43188-6.

26. **Farina, Fabio y Dennunzio, Alberto.** *A Predator-Prey Cellular Automaton with Parasitic Interactions and Environmental Effects.* Milan, Italy : Fundamenta Informaticae, IOS Press, Vol. 83, Number 4, 2008. 0169-2968.

27. **van der Laan, Jan D, Lhotka, Ladislav y Hogeweg, Paulien.** *Sequential Predation: A Multi-model Study.* Czech Republic : Journal of Theoretical Biology. Elsevier, 1995. págs. 149-167. Vol. 174. No. 2. 0022-5193.

28. **J. Futuyma, Douglas y S. Wasserman, Steven.** *Resource Concentration and Herbivory in Oak Forests.* s.l. : Science, 1980. Vol. 210.

29. **Durand-Lose, Jérôme.** *About the Universality of the Billiard ball model.* France : Proceedings Colloquium Mathematical Machines and Computation. Metz, 1998.

30. **Toffoli, Tomasso y Margolus, Norman.** *Celluar Automata Machines: A new environment for modeling.* USA : MIT Press Series in Scientific Computation, 1987. ISBN 0-262-20060-0.

31. **Wolfgang, Reisig.** *Petri Nets - An Introduction.* s.l. : EATCS Monographs on Theoretical Computer Science. Springer Verlag, 1985. Vol. 4.

32. **Naedele, Martin y Janneck, J. W.** *Dessing Patterns in Petri Net System Modelling.* Monterey, CA : Proc. 4th IEEE Int. Conf. on Engineering of Complex Computer Systems, August 1998. págs. 47-54. 0-8186-8597-2.

33. **Gause, G. F.** *The Struggle for Existence.* New York : Hafner, 1934.

34. **Pinheiro da Silva, Paulo y W. Paton, Norman.** *User Interface Modelling with UML.* Amsterdam : Information Modelling and Knowledge Bases XII. IOS Press, 2001. 1-58603-163-5.

35. **Bodart, Francois y Vanderdonckt, Jean.** *Widget Standardisation Through Abstract Interaction Objects.* Istanbul : Advances in Applied Ergonomics. USA Publishing, 1996.

36. **Abud Figueroa, María Antonieta.** *Calidad en la Industria del Software. La Norma ISO-9126.* s.l. : IPN. Revista UPIICSA en Línea. Número 34 Enero - Abril, 2004.

37. **Quatrani, Terry.** *Visual Modelling with Rational Rose 2002 and UML.* s.l. : Addison Wesley, 2002. 0-201-72932-6.

38. **Knuth, Donald E.** *The Art Of Computer Programming.* Boston : Addison-Wesley, 1997. Vol. 2.

39. **Matsumoto, Makoto y Nishimura, Takuji.** *Mersenne Twister: A 623 - dimensionality equidistributed uniform pseudorandom number generator.* s.l. : ACM Transactions on Modeling and Computer Simulation, January 1988. Vol. 8.

40. **Toffoli, Tommaso.** *Lattice - gas vs cellular automata the whole story at last.* Bristol : Old city publishing. Journal of Cellular Automata, 2009. págs. 267-292. Vol. 4.

41. **Gronewold, Anja y Sonnenschein, Michael.** *Event-based modelling of ecological systems with asynchronous cellular automata.* Oldenburg Germany : Ecological Modelling, 1998. págs. 37-52. Vol. 108.

42. **Yen, Hsu-Chun.** *Introduction to Petri Net Theory.* s.l. : Springer. Recent Advances in Formal Languages and Applications, 2006. págs. 343-373. Vol. 25. 978-3-540-33460-6.

Apéndice A. Manual del usuario

El software "CASimulator" permite la simulación de varios autómatas celulares, así como del modelo de depredación secuencial propuesto en este trabajo de tesis. Dependiendo del autómata celular simulado, la aplicación genera la interfaz de usuario necesaria para cada modelo. Si el modelo lo requiere, el usuario es capaz de ingresar los parámetros necesarios para simular diversas condiciones que sean de su interés.

Requerimientos del sistema

A continuación se listan los requerimientos de sistema recomendados para la ejecución de CASimulator:

- Windows XP[2], Windows Vista o Windows 7
- Microsoft .Net Framework 3.5[3]
- Procesador X86 a 1 GHz.
- 256 MB de memoria RAM para la plataforma .Net 3.5
- 30 MB adicionales para la ejecución de CASimulator.
- Tarjeta gráfica con soporte para Direct3D 9 y WDDM (Windows Display Driver Model).

Descripción de la interfaz gráfica

Para iniciar la ejecución del software se debe introducir el CD, navegar a la carpeta "CASimulator" y hacer doble clic en el archivo "CASimulator.exe".

En la Figura 86 se muestra la ventana principal de la aplicación, en la misma es posible distinguir las siguientes secciones:

1) Barra de menú. En ella se incluyen los siguientes menús:
    a) File. Incluye los siguientes sub – menús:
        - Save screenshot. Abre una ventana de diálogo donde el usuario puede guardar una captura de pantalla del estado actual de la simulación.
        - Exit. Esta opción termina la ejecución de la aplicación.
    b) Options. Incluye sub – menús relacionados con la aplicación.
        - Alias. Activa o desactiva la transición suave entre los colores de la lattice.
        - Set model. Abre una ventana de diálogo a través de la cual el usuario puede cambiar el modelo activo.
        - View graph. Muestra la gráfica de comportamiento para el modelo activo.

[2] Debido a las diferencias entre Windows Display Driver Model (WDDM) y Windows XP Display Driver Model (XPDM), la aplicación puede no ser mostrada correctamente en Windows XP.

[3] Al momento de la redacción de este documento es posible descargar gratuitamente la plataforma .Net 3.5 Service Pack 1 desde: http://download.microsoft.com/download/2/0/e/20e90413-712f-438c-988e-fdaa79a8ac3d/dotnetfx35.exe

2) Controles. Incluye los controles necesarios para la manipulación de la simulación.
3) Opciones. Los controles en este grupo contienen opciones relacionadas con la lattice del modelo activo. Los controles presentes en este grupo varían de acuerdo al modelo activo.
4) Parámetros de la simulación. A través de los controles de este grupo, el usuario establece los parámetros pertenecientes al modelo activo. Los controles presentes en este grupo varían según el modelo activo.
5) Barra de estado. La barra de estado proporciona información acerca del número de generaciones que se han simulado. Además incluye un control que permite detener la simulación de acuerdo a un número de generaciones dado.

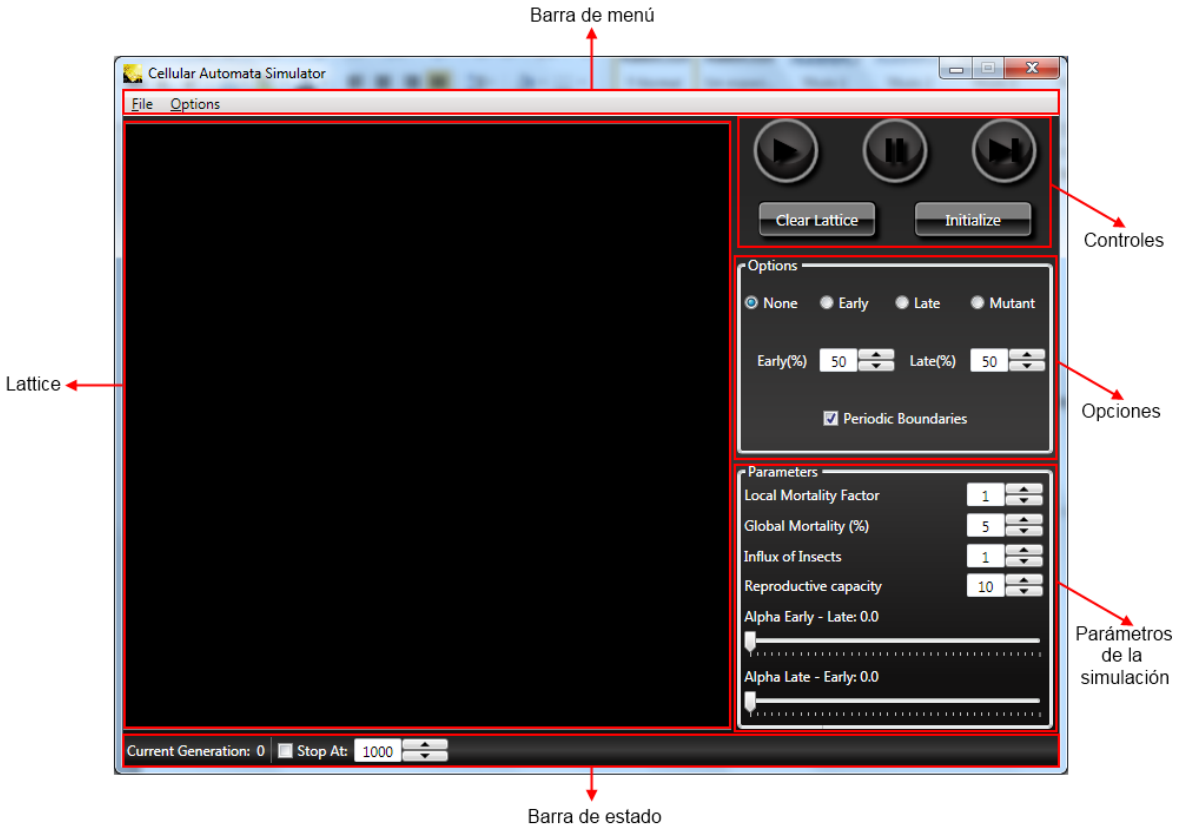


Figura 86. Interfaz gráfica de usuario de CASimulator.

Configuración y ejecución de una Simulación

Los siguientes modelos pueden ser simulados a través de la aplicación:

- Autómatas celulares elementales. Este tipo de modelos fueron estudiados ampliamente por Stephen Wolfram.
- Life. Autómata celular inventado por Conway y popularizado por Martin Gardner en un artículo en la revista Scientific American.
- Diffusion. El modelo HPP propuesto por Hardy, de Passis, y Pomeau, este tipo de autómata fue estudiado a detalle por Toffoli y Margolus.

- Sequential Predation. Objeto de estudio principal en este trabajo de tesis, este modelo ecológico se basa en el estudio de campo descrito en [28].
- Wire World. Autómata celular capaz de simular circuitos lógicos, este modelo fue propuesto por Brian Silverman en 1987. Se popularizo como resultado de su publicación en un artículo en la columna “Computer Recreations” de la revista Scientific American.

No obstante las diferencias entre cada uno de los modelos anteriores, el proceso para configurar y ejecutar una simulación es el mismo. Note que la aplicación inicia con el modelo de depredación secuencial como modelo activo, además cada modelo inicia con una configuración por defecto que permite iniciar una simulación con parámetros establecidos por defecto. En general los pasos necesarios para iniciar una simulación son los siguientes:

1) Establecer el modelo activo.
2) Configurar opciones de la lattice.
3) Configurar parámetros de la simulación.
4) Establecer punto de paro.
5) Iniciar simulación.
6) Desplegar gráfica.
7) Guardar captura de la lattice.

Establecer el modelo activo

Por defecto el modelo activo cuando la ejecución de la aplicación comienza es el modelo de depredación secuencial. Para cambiar este modelo es necesario ir al menú “Options” y hacer clic en el sub – menú “Set Model”. Al hacer esto aparece la ventana de diálogo de la Figura 87. En esta ventana es posible elegir un nuevo modelo, así como establecer el tamaño de la lattice para el mismo. El tamaño máximo permitido es de 512×512 pixeles.

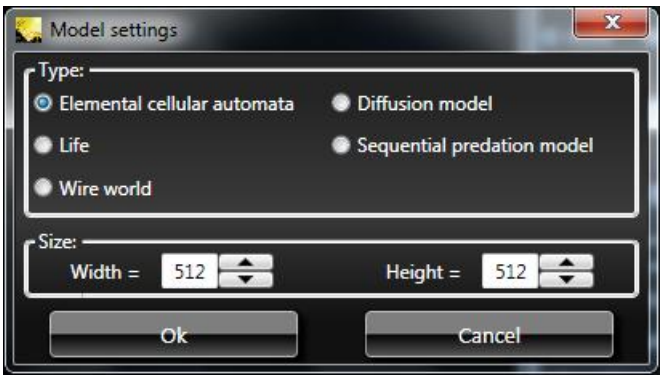


Figura 87. Ventana de diálogo “Model settings”.

Configurar opciones de la lattice

A través del grupo “Options” de la ventana principal (Figura 86), es posible manipular diversos aspectos de la lattice, en general los controles contenidos en este grupo controlan los siguientes aspectos:

a) Configuración manual del estado de un pixel. A través de los "radio – buttons" contenidos en este grupo se puede establecer manualmente el estado de un pixel, el color resultante depende del estado elegido.
b) Distribución inicial de la lattice. Dependiendo del modelo activo, en este grupo existen controles que permiten establecer una distribución aleatoria de la lattice, de acuerdo a porcentajes dados para cada estado permitido.
c) Tipo de fronteras. El "check box" contenido en este grupo permite activar o desactivar las fronteras periódicas en el modelo elegido (excepto para los ACs elementales), cuando se desactiva este control, se establecen condiciones de frontera fija.

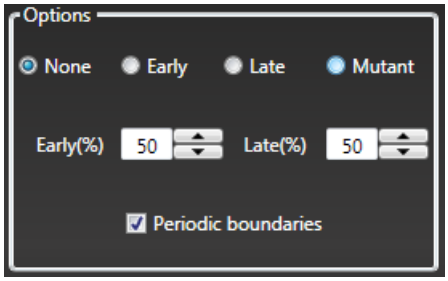


Figura 88. Grupo "Options".

Configurar parámetros de la simulación

En el grupo "Parameters" de la ventana principal (Figura 89) se encuentran los controles necesarios para configurar los parámetros de la simulación, estos controles varían según el modelo activo. Cuando un parámetro es modificado, el cambio se refleja de manera inmediata en la simulación.

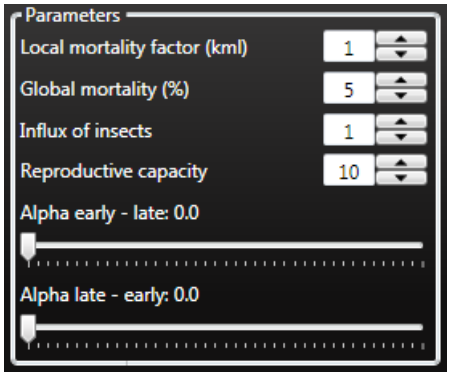


Figura 89. Grupo "Parameters".

Los parámetros configurables cuando el modelo de depredación secuencial está activo son:

- Factor de mortalidad local (k_{ml}). La probabilidad de que un individuo muera como consecuencia de los efectos de la competencia intraespecífica o interespecífica es proporcional a este valor.

- Mortalidad global (M_l). Indica el porcentaje de árboles que son eliminados de manera aleatoria cada temporada.
- Influjo de insectos. Determina el número de insectos que serán introducidos al ecosistema cada temporada.
- Capacidad reproductiva. Indica el número de células dentro de una vecindad de Moore de radio 2 en las cuales es probable que un insecto se reproduzca.
- Coeficientes de competencia interespecífica (α_{e-l} y α_{l-e}). Determinan los efectos inhibitorios entre individuos de distintas especies.

Establecer punto de paro

De manera opcional, es posible indicar la generación donde se desea que la simulación se detenga. En la barra de estado de la aplicación (Figura 90) se pueden observar los controles designados para este propósito. Para explotar esta característica basta con indicar la generación donde se desea detener la simulación, y marcar la casilla de habilitación en la barra de estado. Note que si la generación indicada como punto de paro es menor o igual que la generación actual no podrá iniciar la simulación nuevamente a menos que indique un nuevo punto de paro, o quite la marca de la casilla de habilitación.

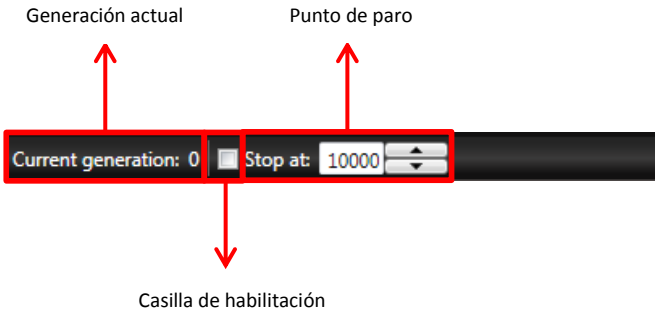


Figura 90. Barra de estado de CASimulator.

Iniciar la simulación

Para iniciar una simulación basta con hacer clic en el botón “Play” de la ventana principal (Figura 91). Además puede detener la simulación haciendo clic en el botón “Pause”, o simular paso a paso con el botón “Step”.

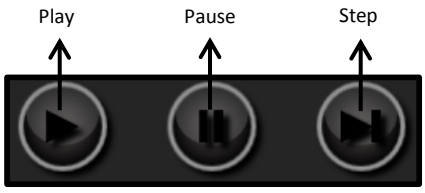


Figura 91. Botones para manipular una simulación.

Desplegar gráfica

Para desplegar la gráfica correspondiente al modelo activo es necesario ir al menú "Options" y hacer clic en el sub – menú "View graph". La visualización de la gráfica puede hacerse sin detener la simulación, sin embargo el rendimiento de la aplicación puede verse afectado debido a esto.

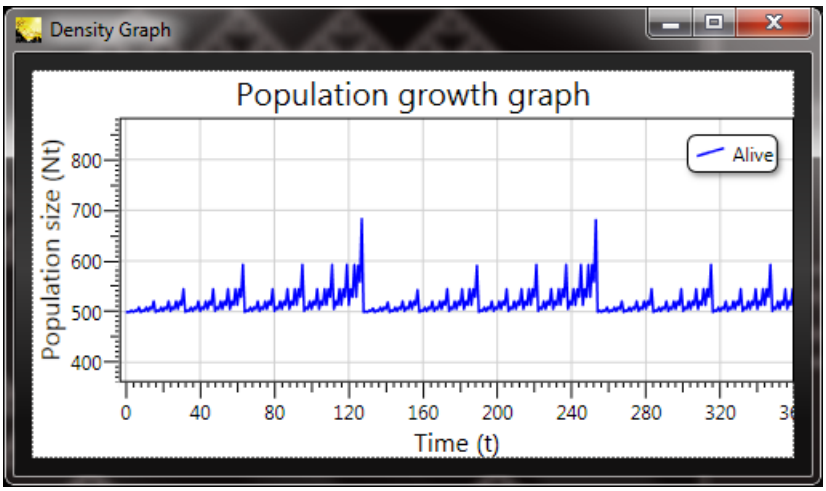


Figura 92. Ventana de gráfica.

Guardar capturas de la lattice

Para guardar una captura de la lattice, es necesario hacer clic en el menú "File" y posteriormente acceder al sub – menú "Save screenshot", al hacer esto aparece una ventana de diálogo (Figura 93) donde se puede seleccionar la ruta y el nombre de la captura. Las capturas son almacenadas en el formato PNG.

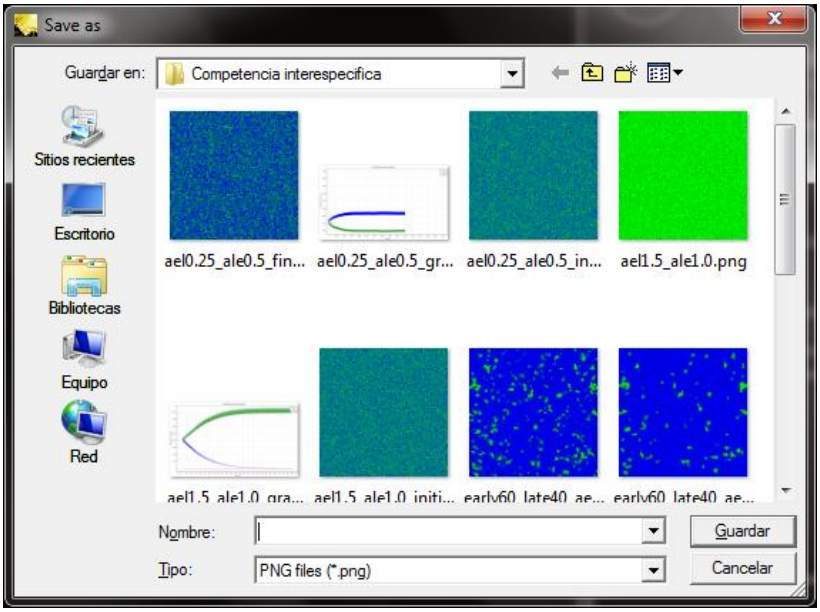


Figura 93. Ventana de diálogo "Save as".