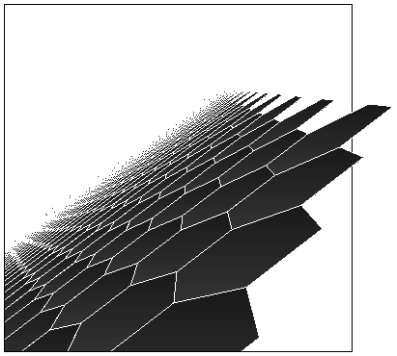
Cellular Systems



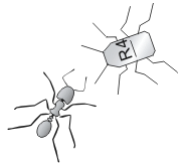
Modeling cellular systems

We want to define the simplest nontrivial model of a cellular system. We base our model on the following concepts:

- *Cell* and *cellular space*
- *Neighborhood* (local interaction)
- Cell *state*
- *Transition rule*

We do not model all the details and characteristics of biological multicellular organisms but we obtain simple models where many interesting phenomena can still be observed

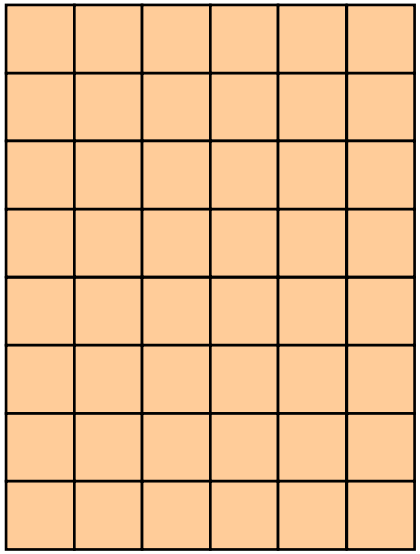
- There are many kinds of cellular system models based on these concepts
- The simplest model is called *Cellular Automaton* (CA)

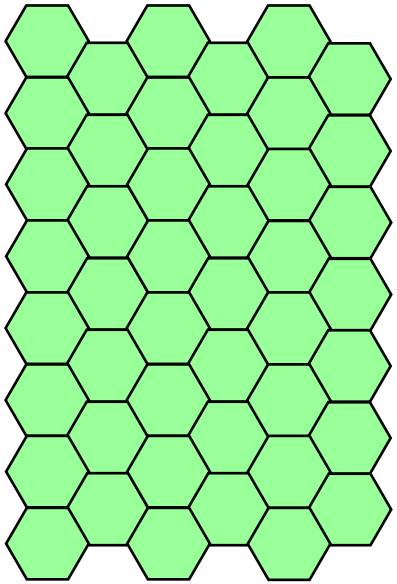


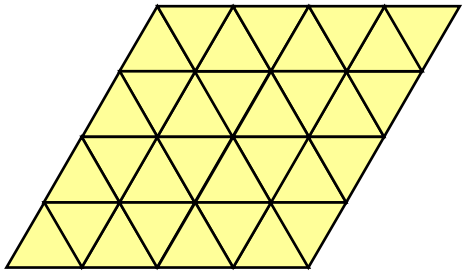
Cellular space

1D

2D

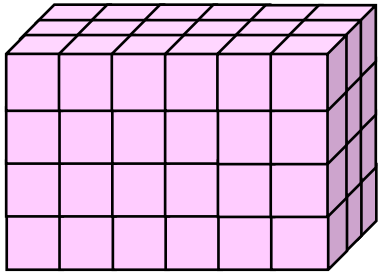






...

3D

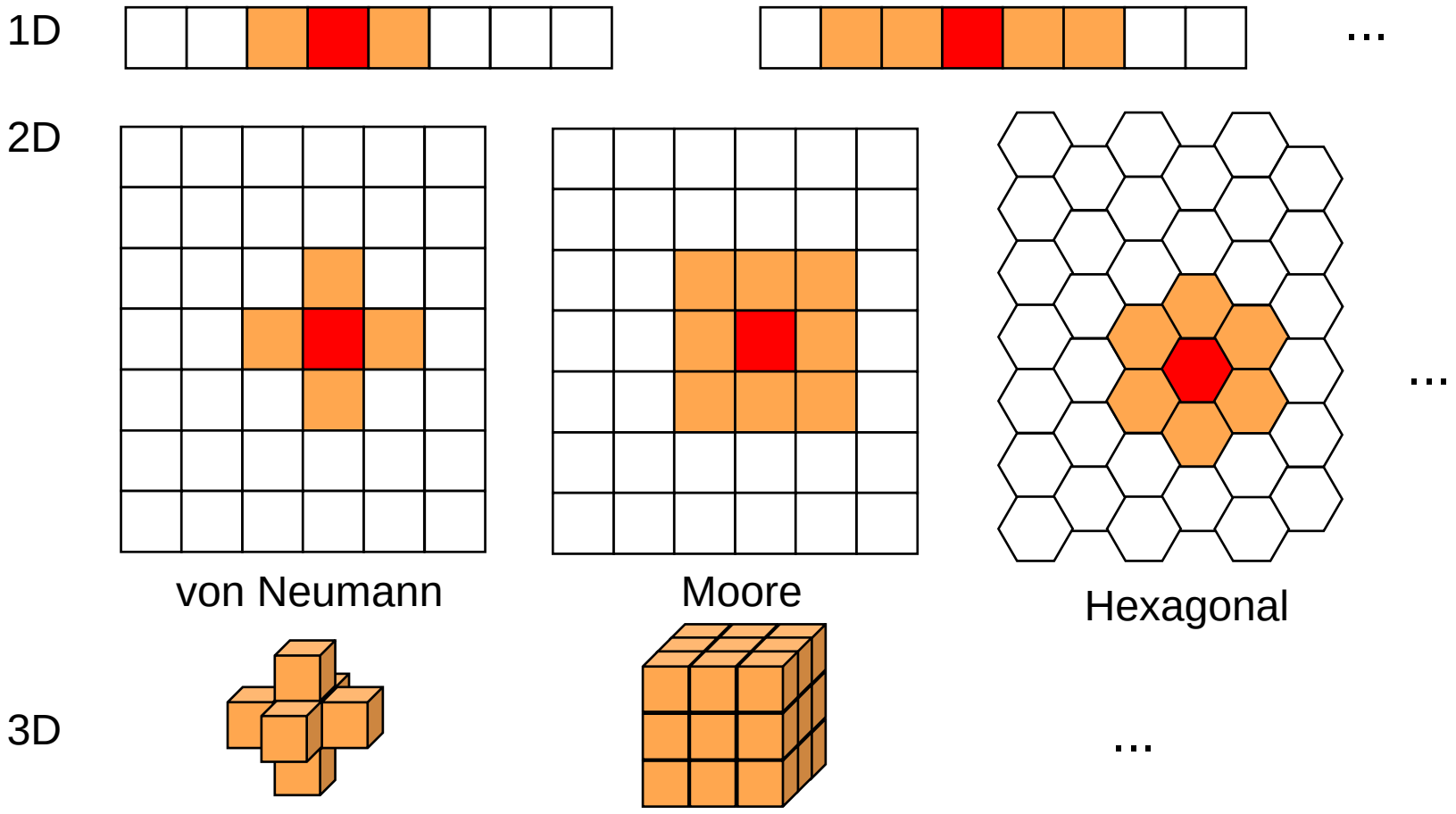


...

and beyond...

Neighborhood

- Informally, it is the set of cells that can influence *directly* a given cell
- In *homogeneous* cellular models it has the same shape for all cells



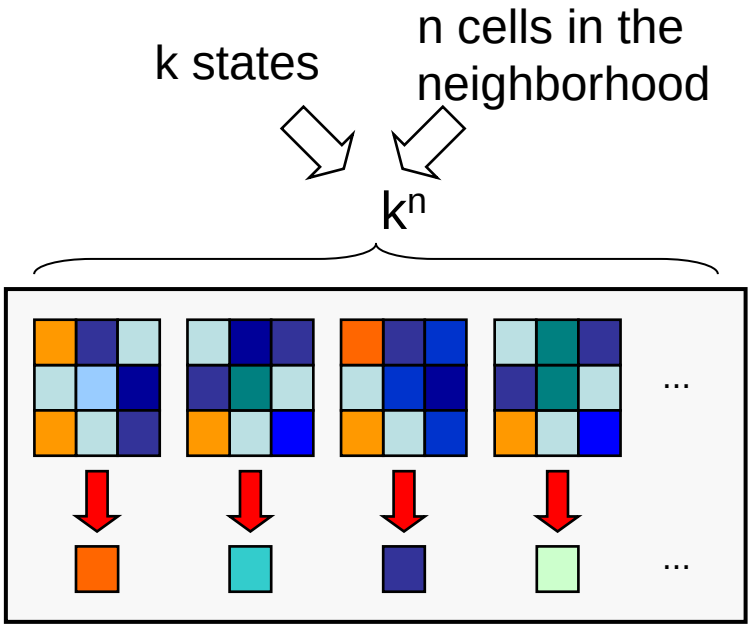
State Set and Transition Rule

The value of the **state** of each cell belong to a finite set, whose elements we can assume as being numbers. The value of the state is often represented by cell colors. There can be a special **quiescent state** s_0.

$$S = \{s_0, \ldots, s_{k-1}\} = \{0, \ldots, k-1\} = \{\bullet, \ldots, \bullet\}$$

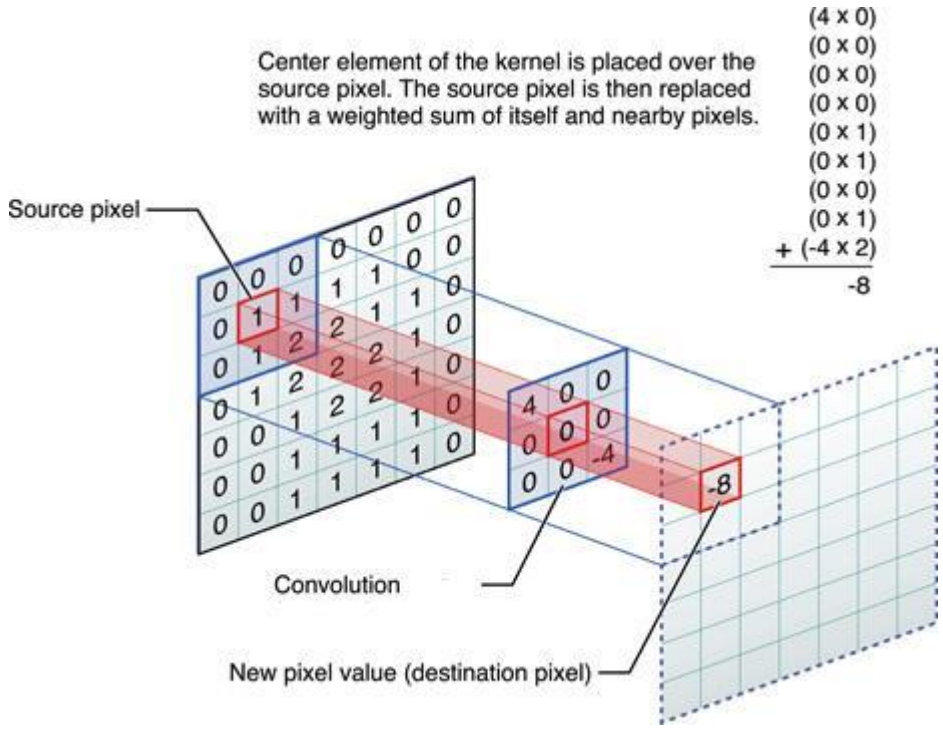
The **transition rule** is the fundamental element of the CA. It must specify the new state corresponding to each possible configuration of states of the cells in the neighborhood.

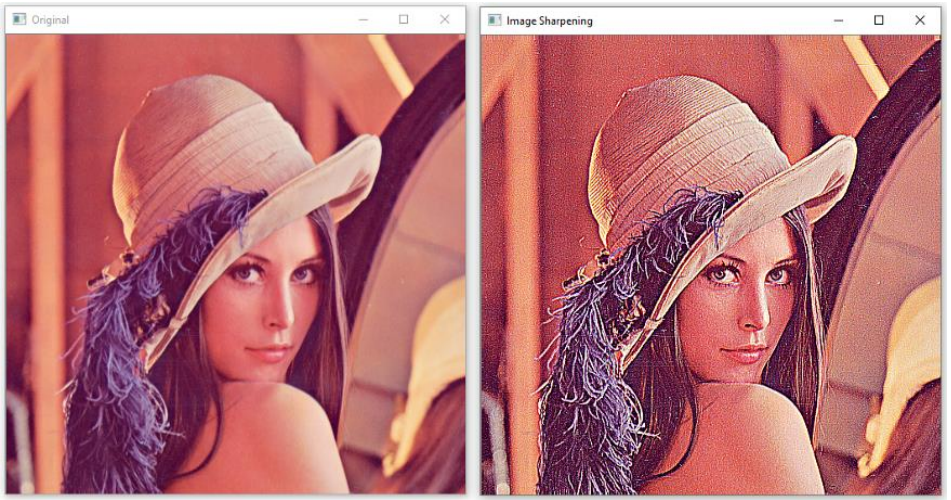
The transition rule can be represented as a **transition table**, although this becomes rapidly impractical.

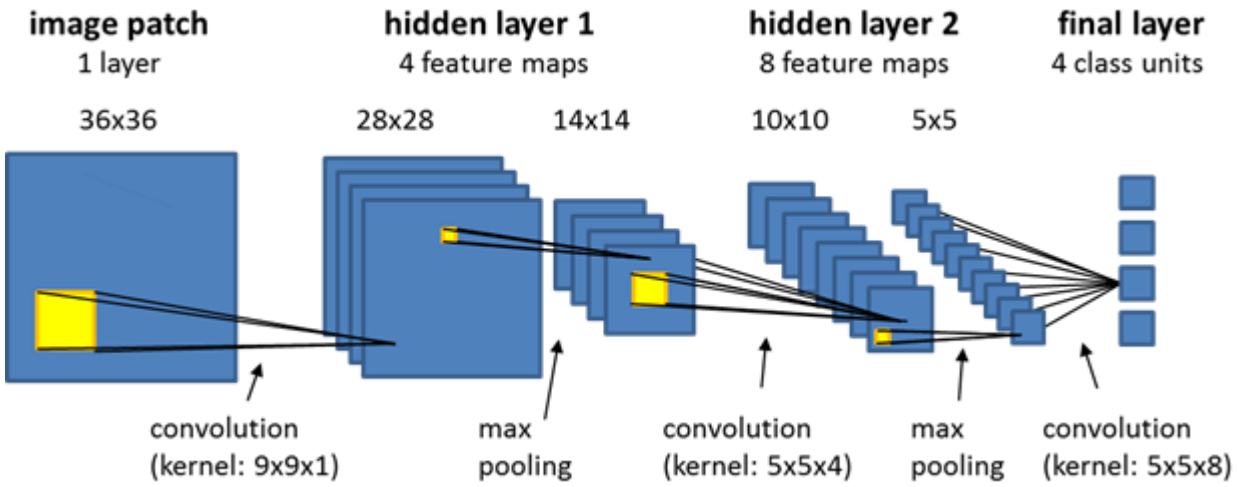


transition table

Convolution

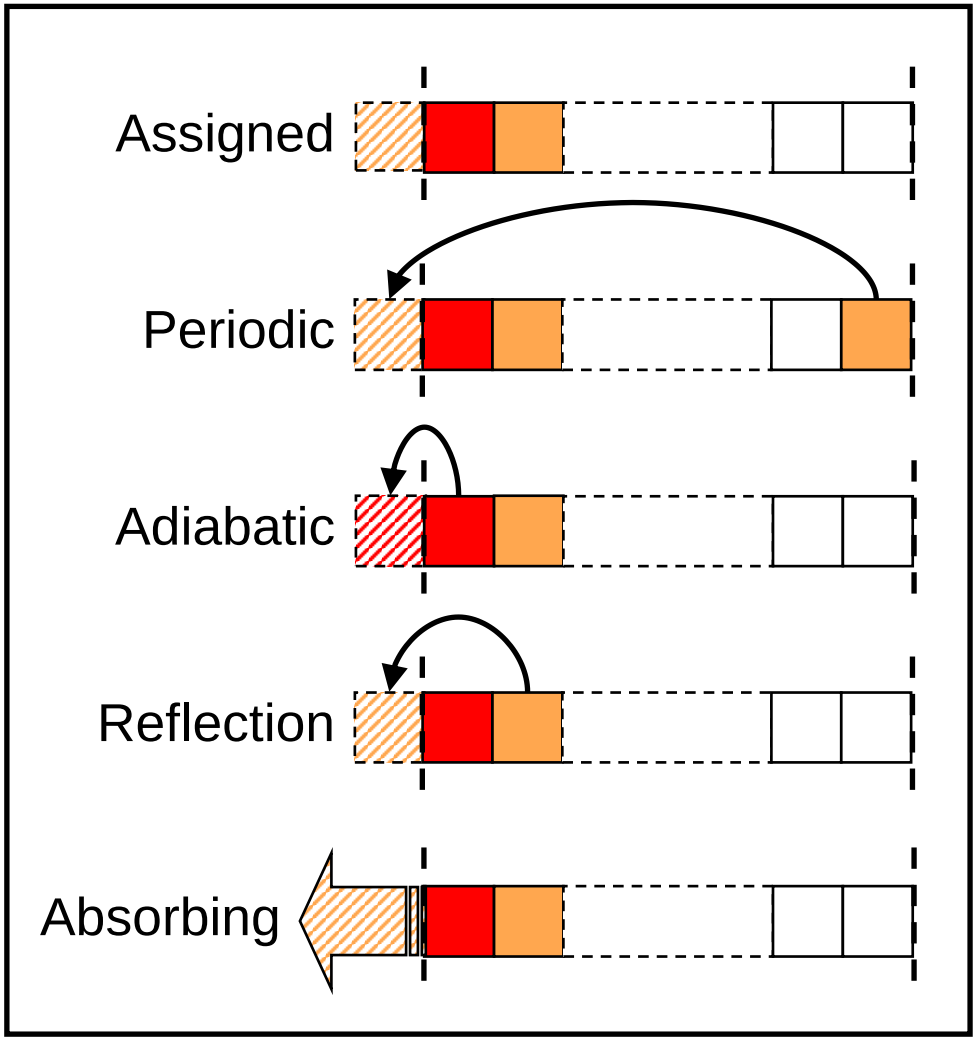






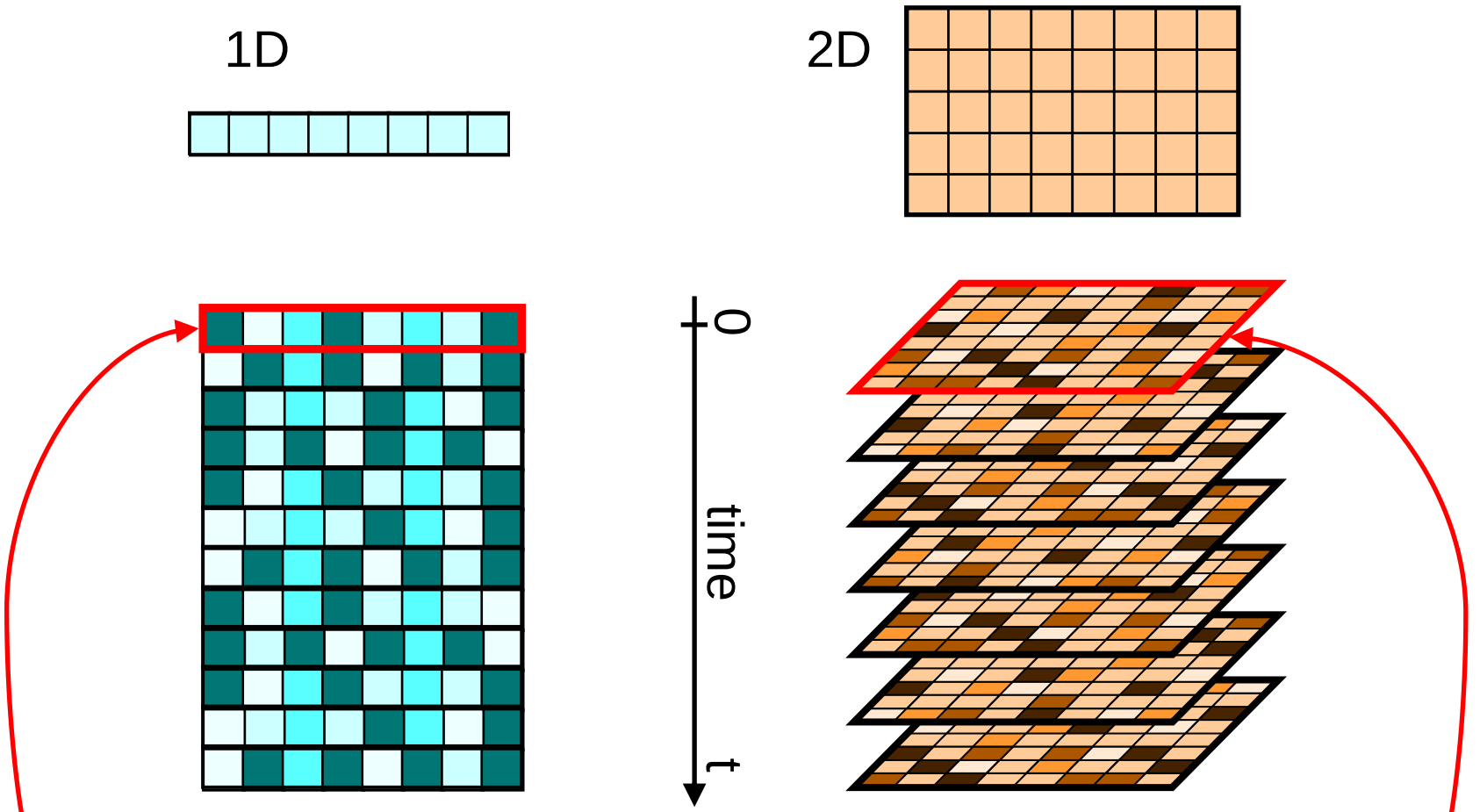
Boundary Conditions

- If the cellular space has a boundary, cells on the boundary may lack the cells required to form the prescribed neighborhood
- *Boundary conditions* specify how to build a "virtual" neighborhood for boundary cells



Some common kinds of boundary conditions

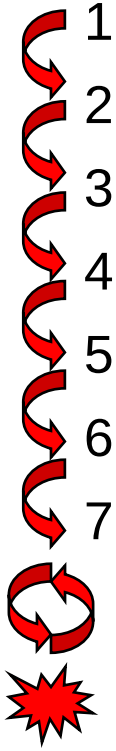
Initial Conditions



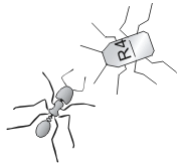
In order to start with the updating of the cells of the CA we must specify the initial state of the cells (**initial conditions** or **seed**)

In practice...

To implement and run a CA experiment



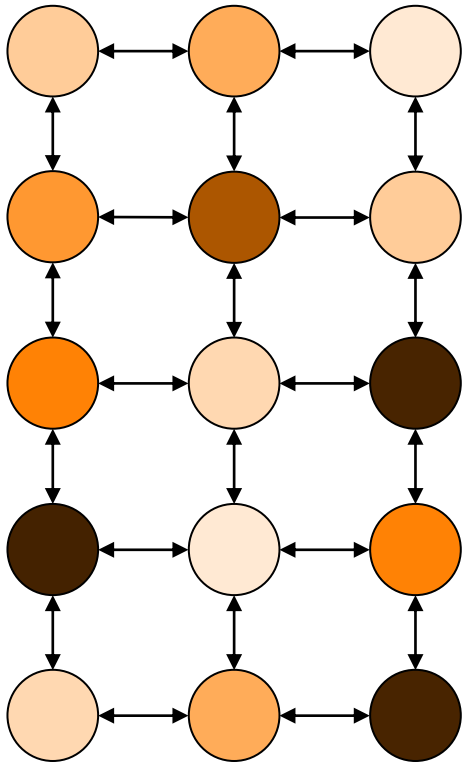
1. Assign the geometry of the CA space
2. Assign the geometry of the neighborhood
3. Define the set of states of the cells
4. Assign the transition rule
5. Assign the boundary conditions
6. Assign the initial conditions of the CA
7. Repeatedly update all the cells of the CA, until some stopping condition is met (for example, a pre-assigned number of steps is attained, or the CA is in a quiescent state, or cycles in a loop,...).



Informal definition of CA

A Cellular Automaton is

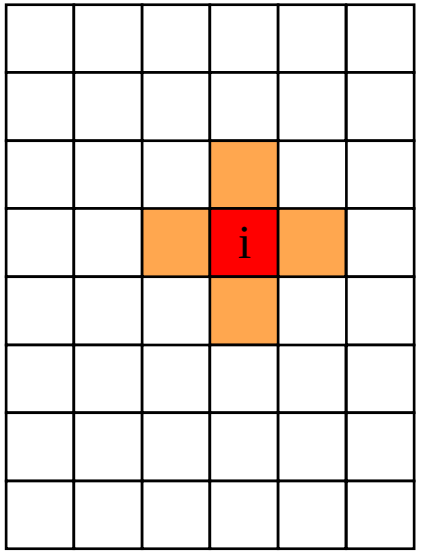
- a **geometrically structured** and
- **discrete** collection of
- **identical** (simple) systems called **cells**
- that **interact** only **locally**
- with each cell having a local **state** (memory) that can take a **finite** number of values
- and a (simple) **rule** used to **update** the state of all cells
- at **discrete time** steps
- and **synchronously** for all the cells of the automaton (global “signal”)



Formal definition of CA

A Cellular Automaton is

- an **n-dimensional lattice** of
- **identical** and **synchronous** finite state machines
- whose state s is updated (synchronously) following a **transition function** (or transition rule) ϕ
- that takes into account the state of the machines belonging to a **neighborhood** N of the machine, and whose geometry is the same for all machines



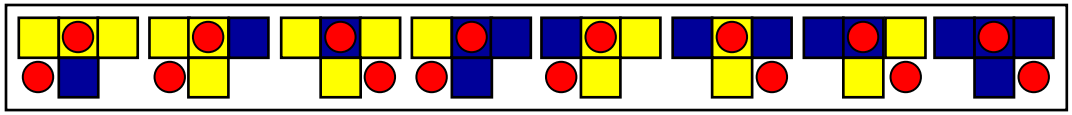
$$s_i(t+1) = \phi(\, s_j(t) \,;\, j \in N_i \,)$$

Variants and Extensions

The basic CA is discrete in space, time and state; updates all its cells synchronously; uses the same neighborhood geometry and transition rule for all cells.

We can relinquish some of these prescriptions and obtain:

- **Asynchronous** CA (for example, *mobile automata*, where only one cell is active at each time step, and the transition rule specifies the fate of the activation)



- **Non-homogenous** (or non-uniform) CA
- Continuous-state CA (**Coupled Map Lattices**): Lattice Boltzmann models
- Continuous-state and time CA (**Cellular Neural Networks**): EDP

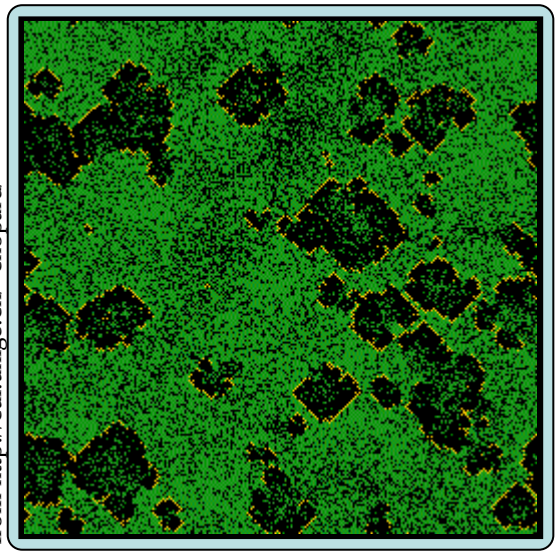
Probabilistic (Stochastic) CA

So far we have considered only **deterministic** CA.

To model many phenomena it is useful to transition rules that depending on some externally assigned probability

Example: The **forest fire model**

- Each cell contains a green tree, a burning tree, or is empty
  - A burning tree becomes an empty cell
  - A green tree with at least a burning neighbor becomes a burning tree
  - A green tree without burning neighbors becomes a burning tree with probability *f* (probability of lightning)
  - An empty cell grows a green tree with probability *g* (probability of growth)

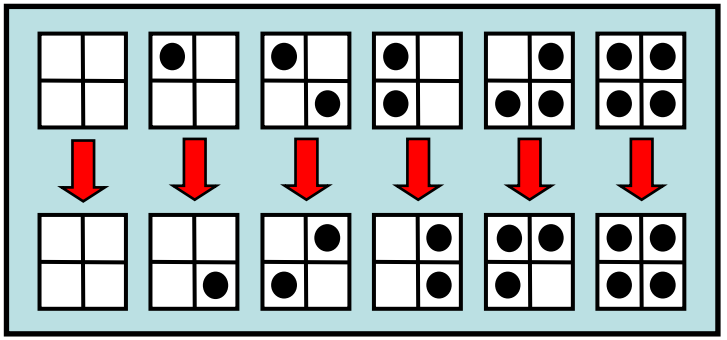


from http://cui.unige.ch/~chopard/

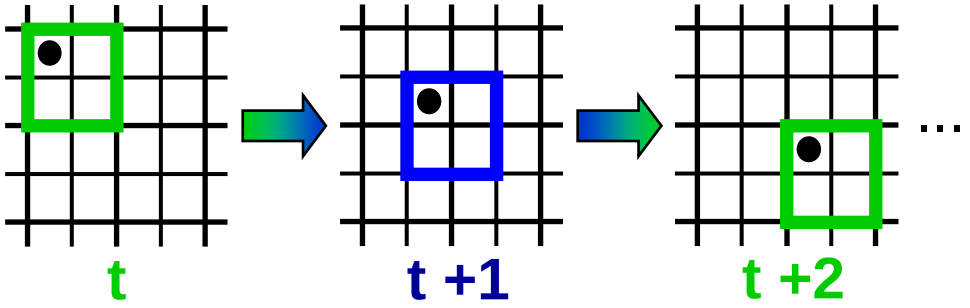
The parameters can be varied in a continuous range and introduce some “continuity” in the discrete world of CA models

Particle CA

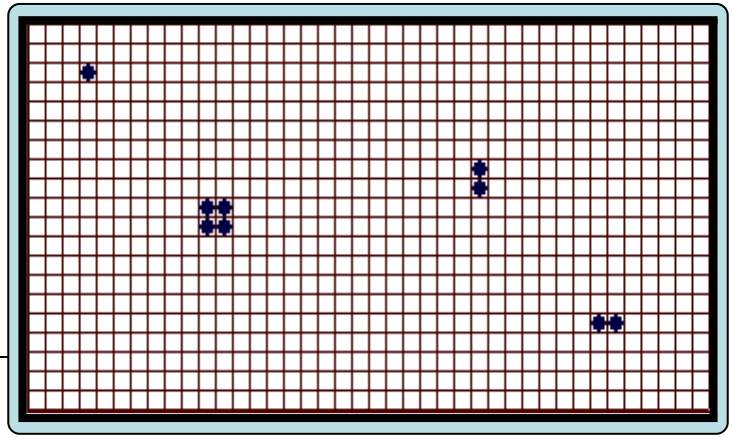
CA can be used to model phenomena that involve particles. The transition rule can be specified in terms of the motion of particles within **blocks** of two by two cells (block rules).



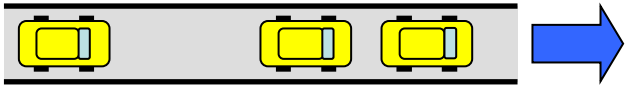
The automaton space is **partitioned** in non-overlapping blocks



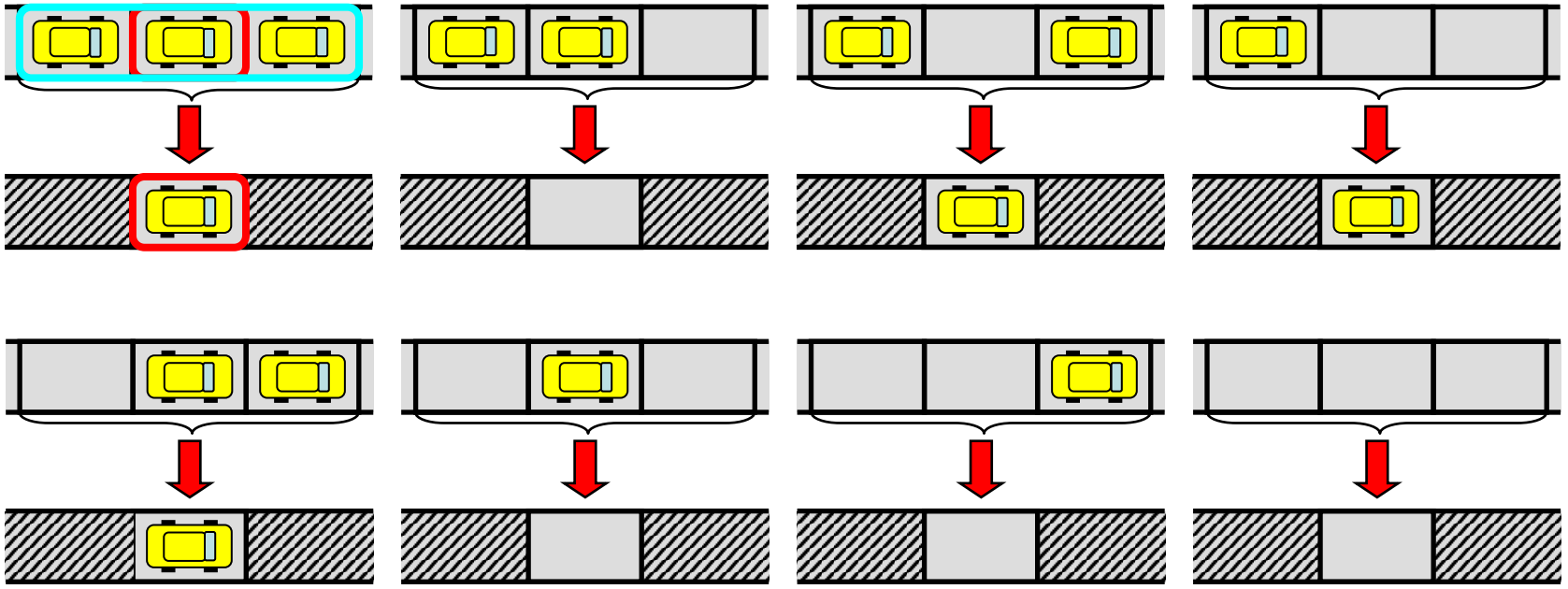
To allow the propagation of information the position of the blocks alternates between an odd and an even partition of the space (Margolus neighborhood). VIDEO



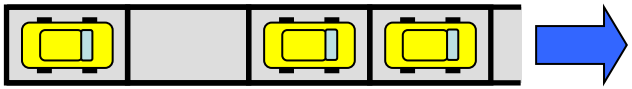
Example: Modeling Traffic



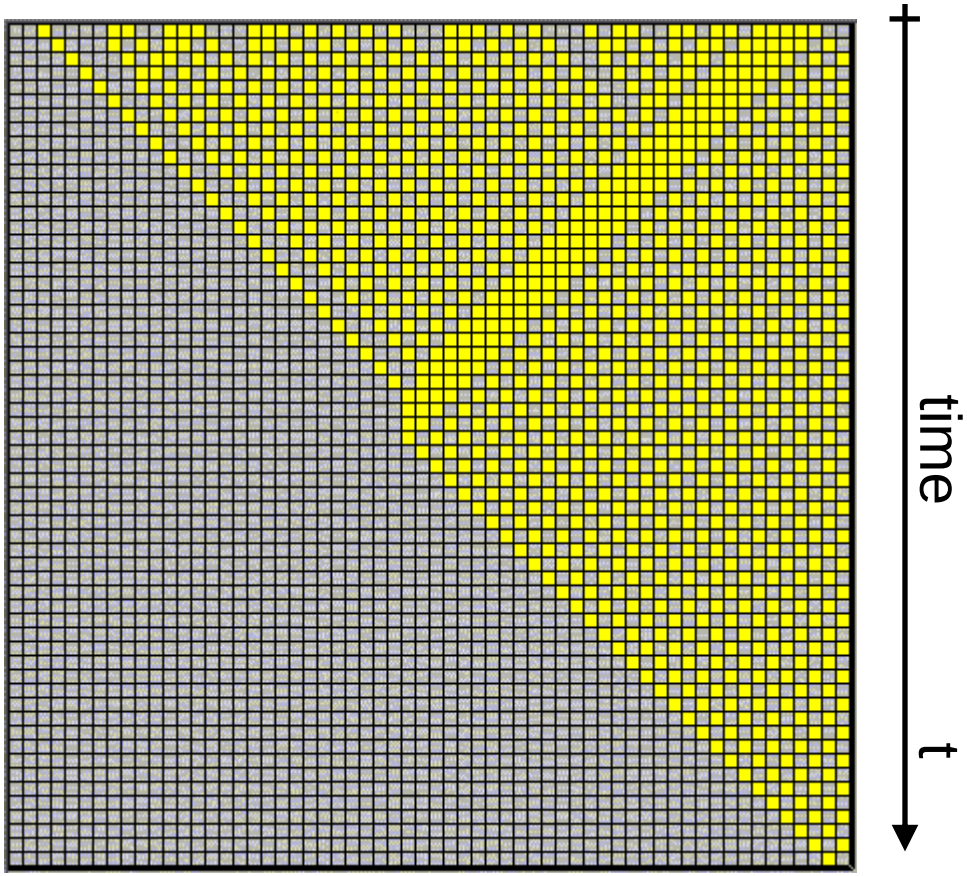
We construct an elementary model of car motion in a single lane, based only on the **local** traffic conditions. The cars advance at discrete time steps and at discrete space intervals. A car can advance (and must advance) only if the destination interval is free.



Example: Traffic Jam



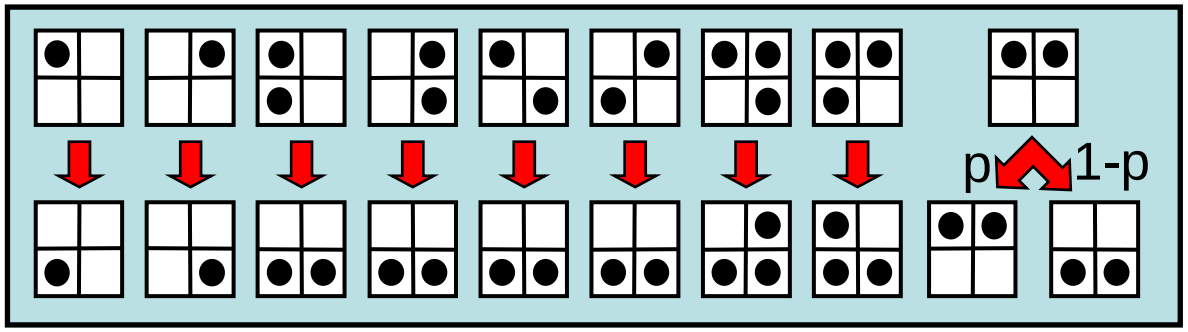
Running the *traffic CA* with a high-density random initial distribution of cars we observe a phenomenon of backward propagation of a region of extreme traffic congestion (traffic jam).

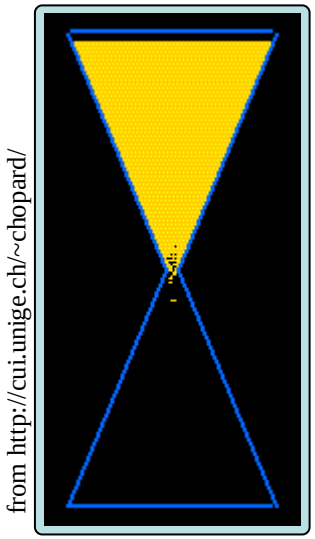


Complex Systems

Cellular systems allow the modeling and simulation of phenomena that are difficult to describe with conventional mathematical techniques

Example: The sand rule with friction





from http://cui.unige.ch/~chopard/

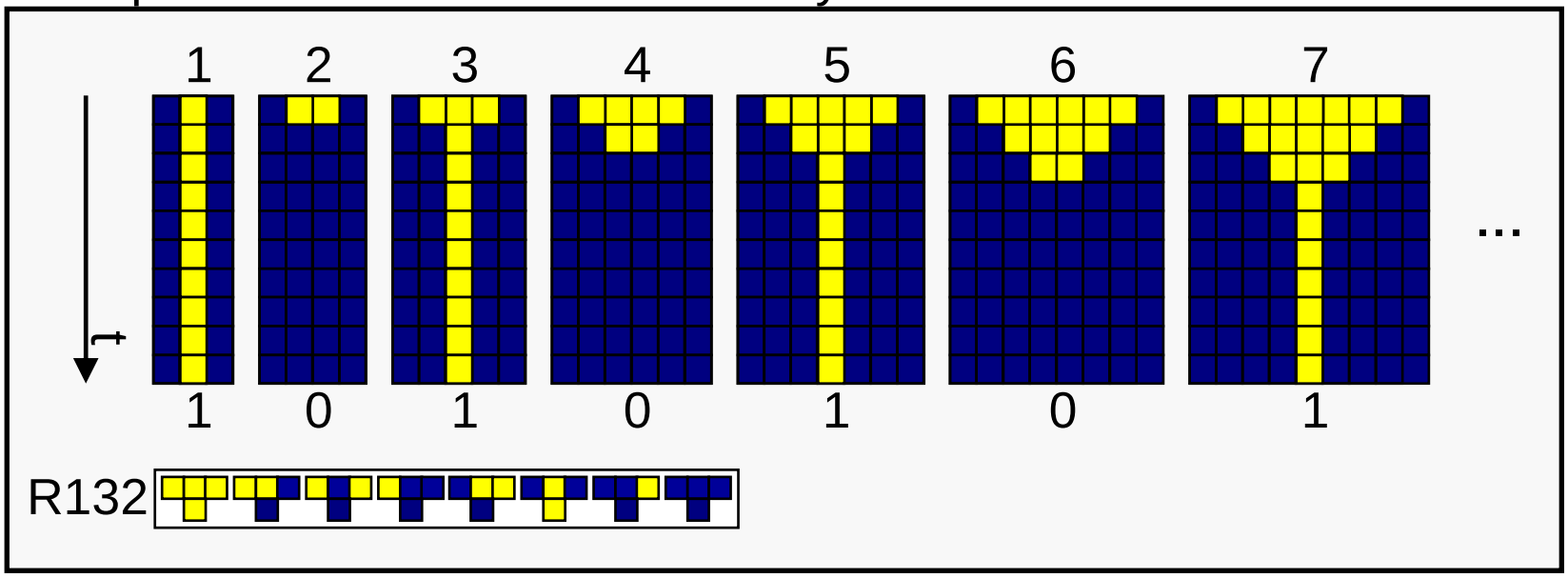
This kind of model permits the exploration of the behavior of *granular media*, which is difficult with conventional tools (e.g., PDEs)

Video

Computation with CA

CA used as input-output devices. The initial state is the input. The CA should go to a quiescent state (fixed point), which is the output.

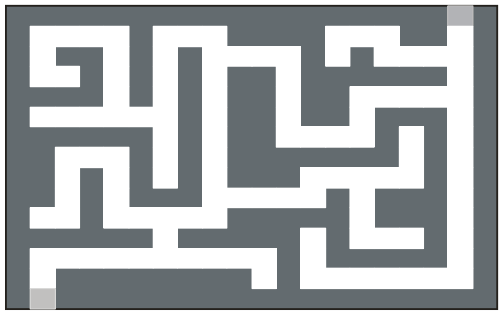
Example: Remainder after division by 2



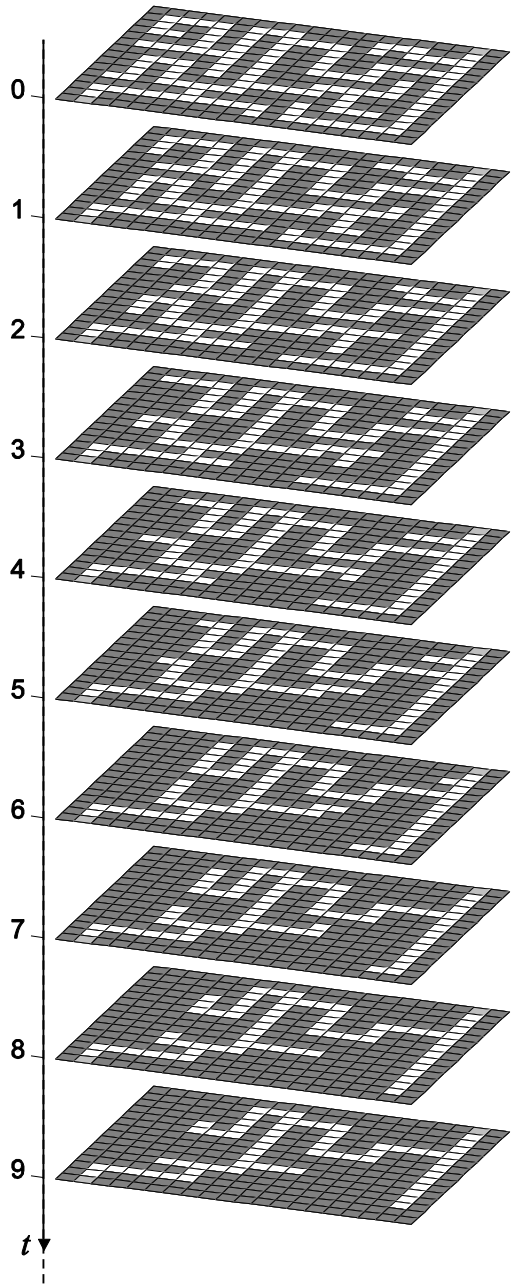
The difficulty stems from the fact that we use a local rule to evaluate a property that depends on information distributed globally.

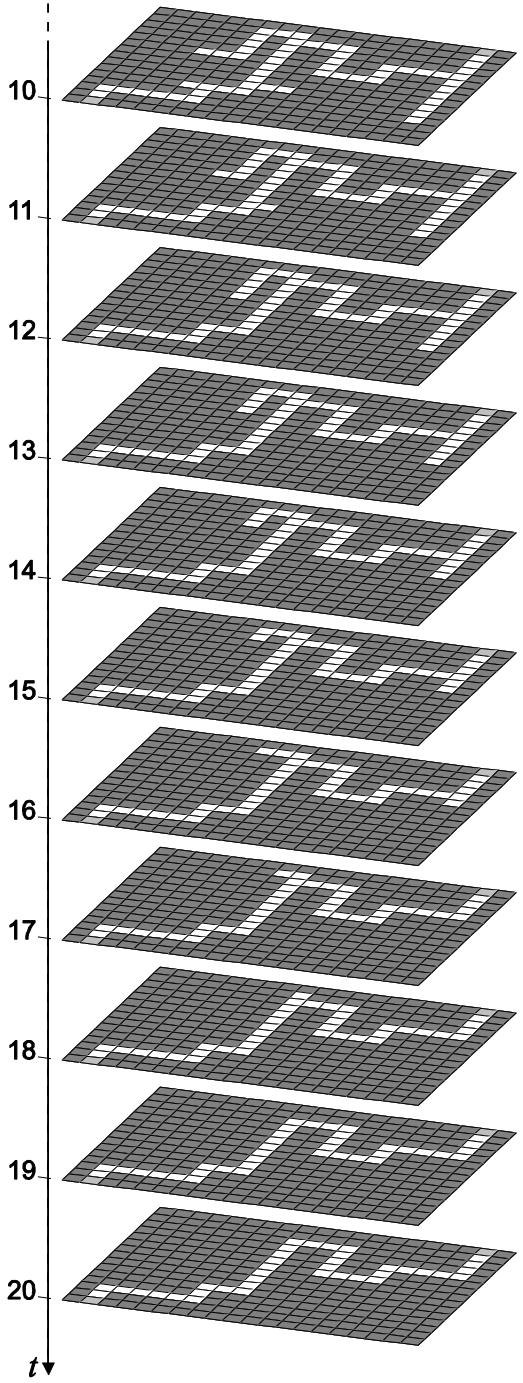
No existe una metodología general para inferir las reglas!!!!

Example: CA maze solver



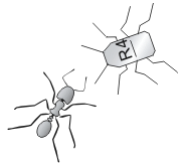
- Given a maze the problem consists in finding a path from the entrance to the exit.
- The conventional approach marks blind alleys sequentially
- The CA solver removes blind alleys in parallel.
- Von Neumann





Computational Universality

- In *Life* we can define signals (as streams of gliders interpreted as bits), implement all logic gates (AND, NOT,...), implement delays, memory banks, signal duplicators, and so on.
- Hence, *Life* can emulate any computing machine; we say that it is capable of **universal computation**.
- The theory of computation says that, in general, given an initial state for the automaton, there is no short-cut way to predict the result of *Life*'s evolution. We must run it.
- We say that *Life* is **computationally irreducible**.
- In simple words, this means that a very simple CA such as *Life* (and Rule 110 in 1D) can produce highly nontrivial behaviors, that cannot be predicted simply by observing the transition rule.
- The "universe" constituted by a CA can be an interesting backcloth for the emergence of complex phenomena.



Computational Universality

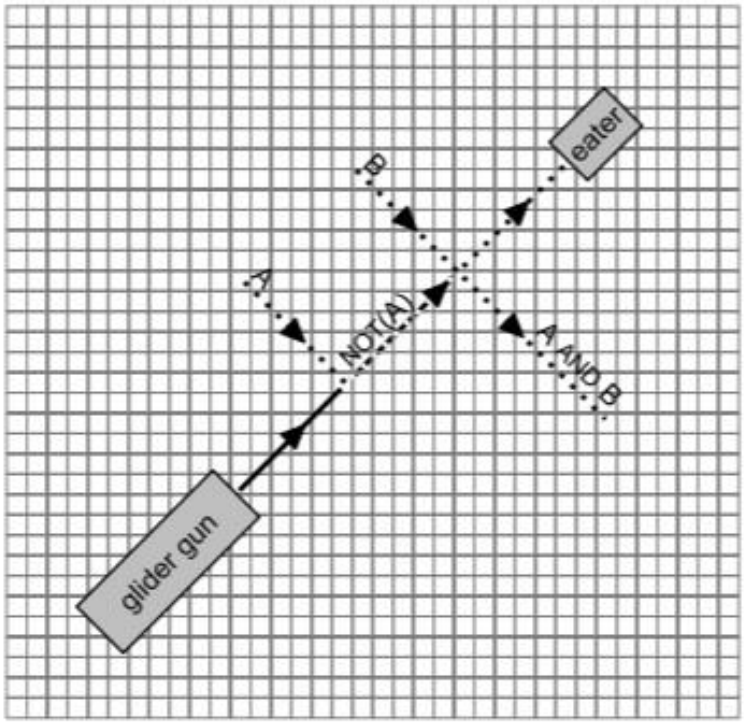
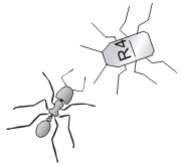
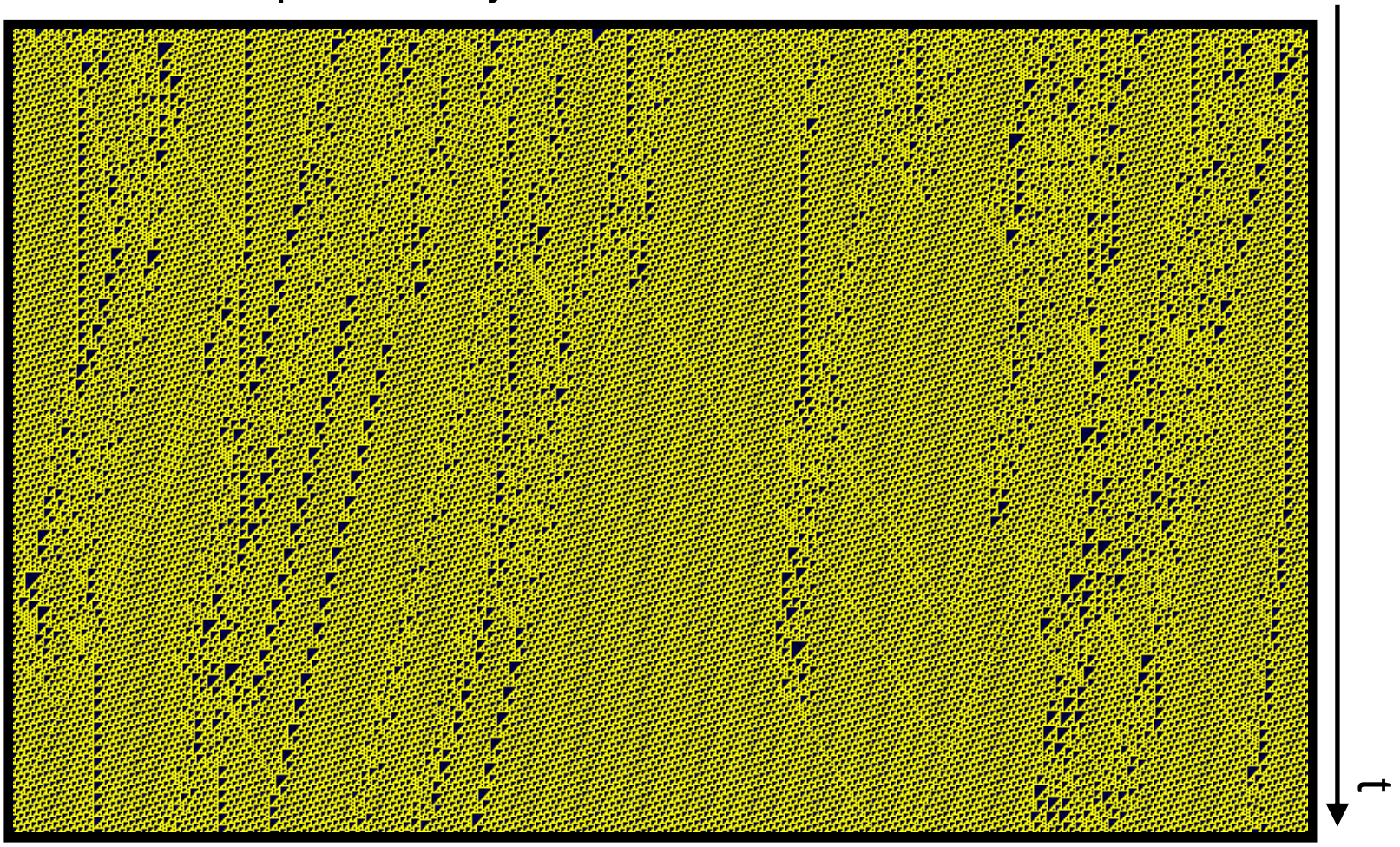


Figure 2.25 A schematic representation of the implementation of a logic gate in Conway's Life CA. The binary signals A and B correspond to streams of gliders (the presence of a glider in the stream corresponds to a logic 1, and its absence to a logic 0). The gliders produced by the glider gun are annihilated when they crash into a glider belonging to A, so that a stream corresponding to NOT(A) emerges from the interaction. The same kind of interaction between the streams B and NOT(A) produce a stream corresponding to A AND B and some residual gliders that are annihilated by the eater.



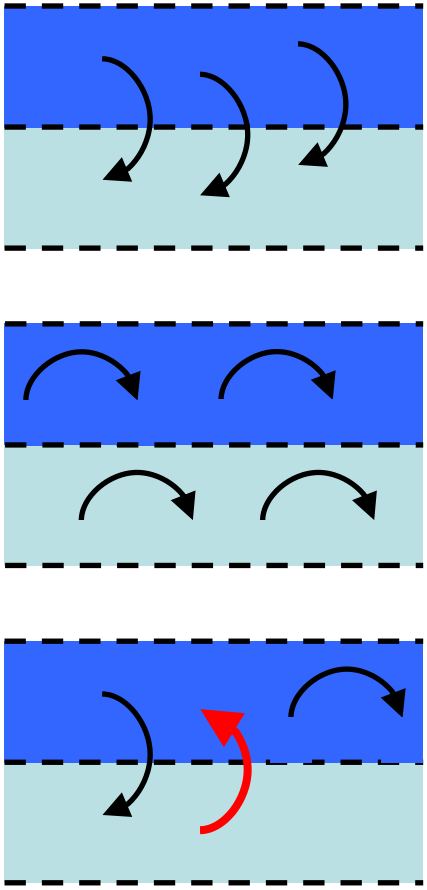
Universality in 1D CA

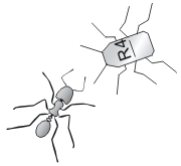
CA even simpler than *Life* display the same properties.
Rule 110 is computationally universal.



The Growth of Complexity

- Usually a machine produces machines les complex than itself: can we prove formally that there exist machines which can produce more complex machines?
- von Neumann's approach:
  - A machine capable of self-reproduction would produce machines of equal complexity
  - If the self-reproduction process could tolerate some "error" (*robust* self-reproduction) then some of the resulting machines might have greater complexity than the original one: mutation



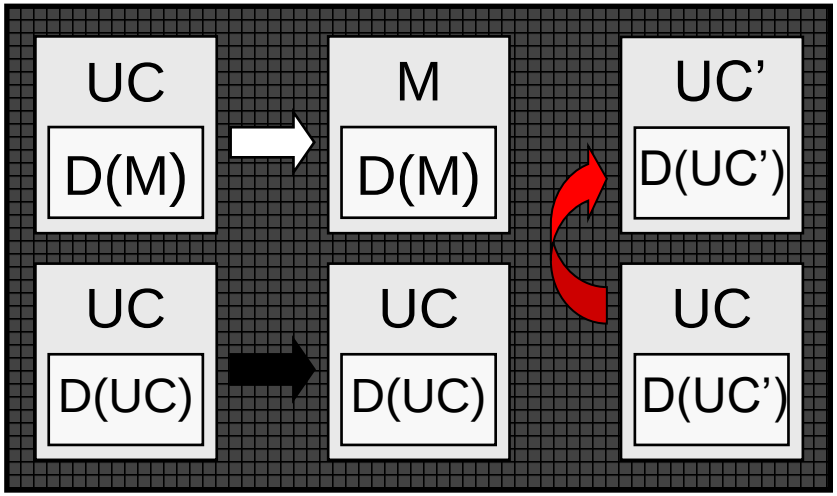


Self-Reproducing Automata

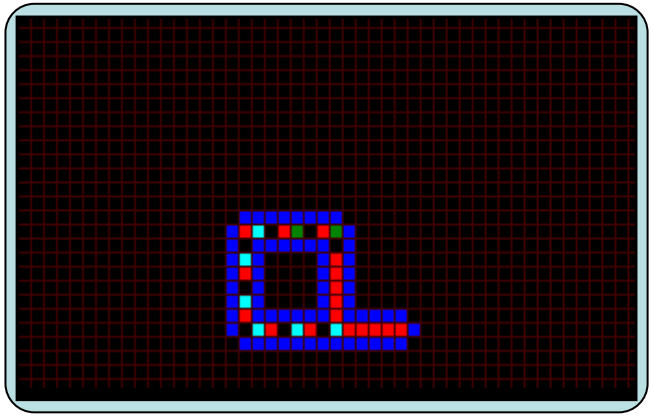
von Neumann solved his problem by defining an automaton composed by a **universal constructor** UC and a description D(M) of the machine to be generated.

von Neumann automata is quite complex (29 states per cell, and about 200.000 active cells)

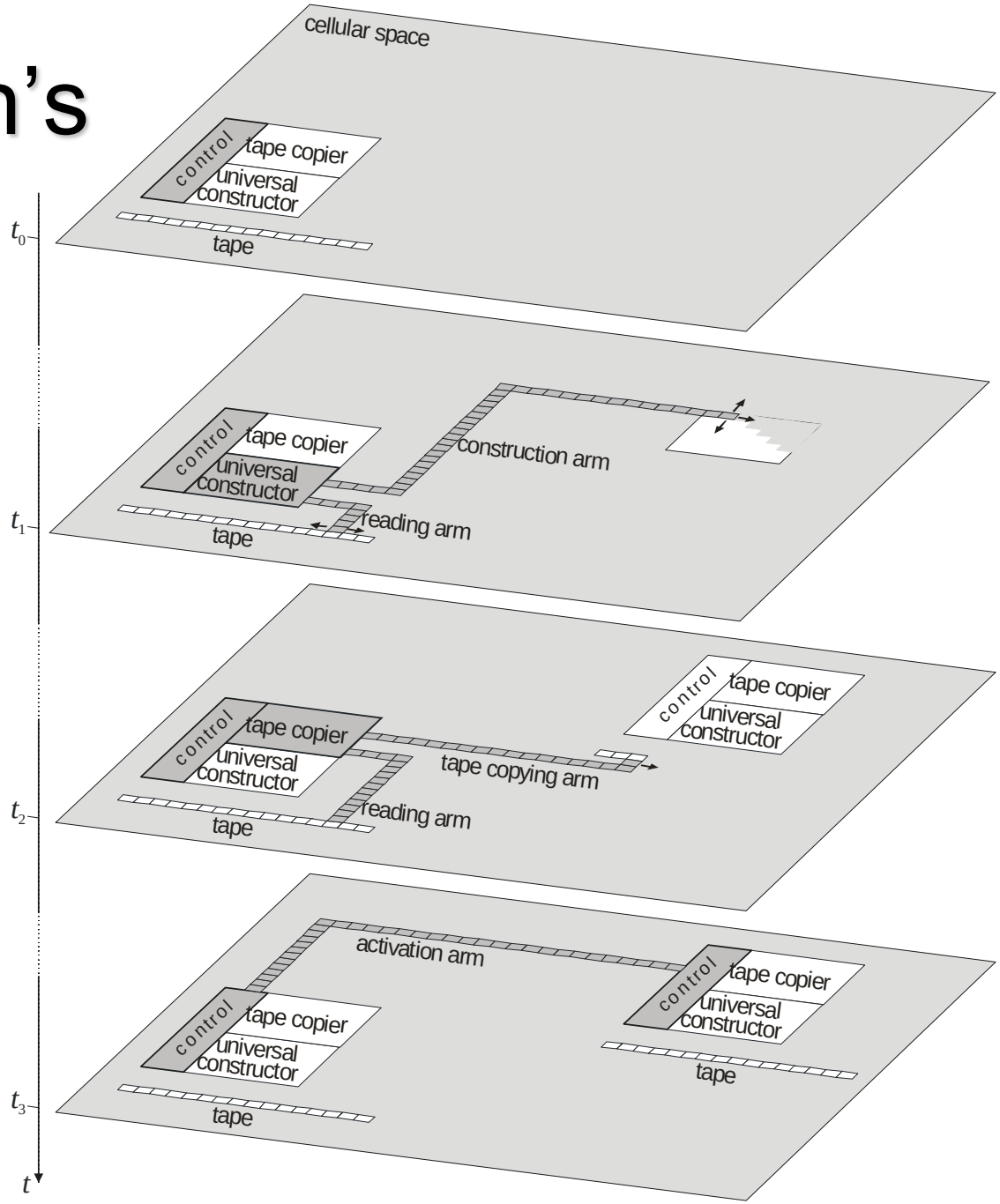
Other scientists focused on the issue of self-reproduction and offered simpler solutions to this sub-problem (trivial self-reproduction). VIDEO

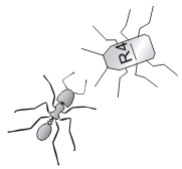


Example: Langton's Loop

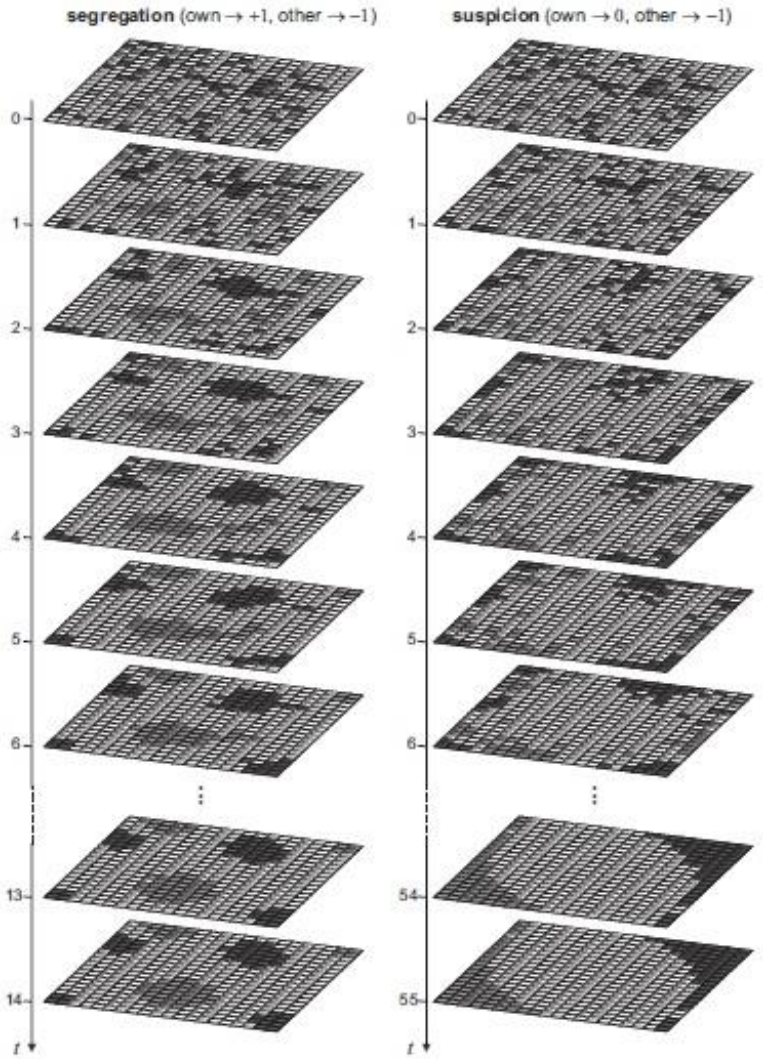


von Neumann's Automaton





Sistemas complejos



gure 2.30 Two examples of a space-time diagram obtained with the CA for the nulation of social dynamics considered by Sakoda (1971). Cells represented as light d dark gray correspond to agents belonging to two distinct groups of 50 individuals ch. Initially, the agents are distributed randomly on a cellular space of 20 × 20 cells th impenetrable boundaries. The empty positions are represented as white cells. e the text for the description of the CA rules and dynamics.

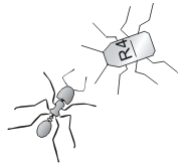
Analysis and Synthesis

Both analysis and synthesis of cellular systems are usually difficult problems. The problem is once again the fact that the link between the local rules and the global behavior is not obvious. A number of different techniques are used.

Analysis: Phenomenological approach; Dynamical System Theory (attractors, cycles...); Analytic approach (global mapping, algebraic properties,...). Statistical Mechanics concepts; Probabilistic approach...

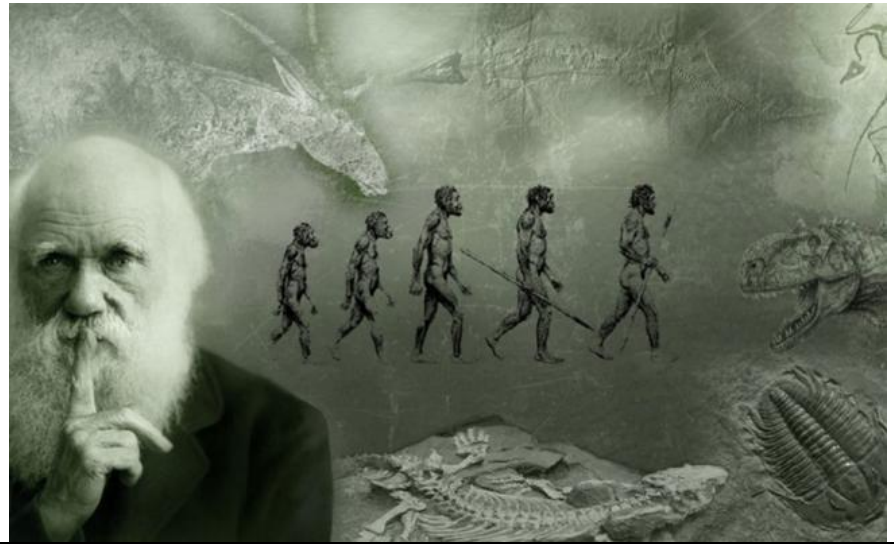
Synthesis: By hand (e.g., *Life*'s zoo); Based on some idea about a possible underlying "microscopic" process...

Due to the absence of general principles to rules producing a desired global behavior, the synthesis of cellular systems is a field particularly suited to the application of **Evolutionary Techniques**.

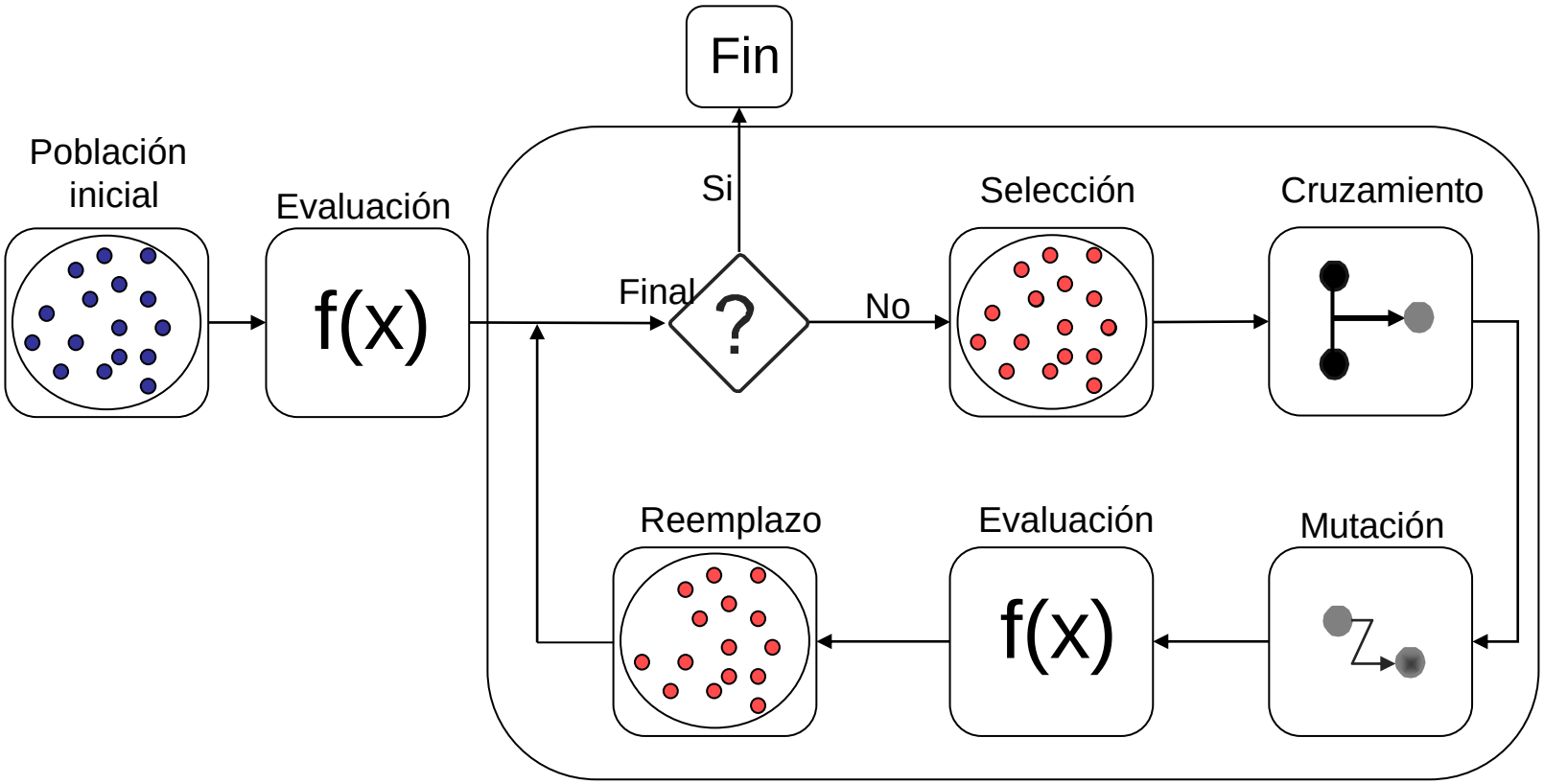


Algoritmos Evolutivos

- Familia de metaheurísticas poblacionales
- Inspirados en el proceso natural de la evolución de las especies de Darwin
- Guiados por el principio de supervivencia del más apto
- Usan ideas de selección natural, reproducción y diversidad genética
- Basado en la aplicación probabilística de operadores

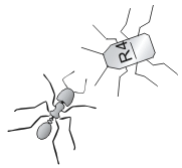


Algoritmos Genéticos



Computación/comportamiento emergente

- Se entiende por comportamiento emergente al que surge de la acción de componentes simples con información y comunicación limitadas, siendo capaz de procesar informacion global.
  - Ejemplos: colonias de insectos, sistemas económicos, el sistema inmunológico, el cerebro, etc.
- Ideales para sistemas descentralizados.
- Los sistemas de computación emergente son robustos, rápidos y adaptables para el procesamiento de información.

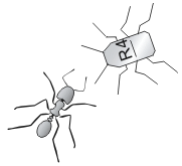


Síntesis de Autómatas Celulares

- Los autómatas celulares se inician en una configuración compuesta por 0s y 1s, que cambia en pasos de tiempo discretos donde se modifican todas las celdas en forma simultánea, de acuerdo a un conjunto de reglas de transición.

- Las reglas de un autómata celular pueden ser expresadas como una tabla que define para cada vecindad el valor con el que debe ser actualizada la celda considerada.

- En el caso del ejemplo presentado en la Figura 6 se utiliza la regla de la mayoría: para cada vecindad de tres celdas adyacentes, el nuevo valor es decidido por la mayoría de la votación de las tres celdas.



Síntesis de Autómatas Celulares

Rule table:

neighborhood:	000	001	010	011	100	101	110	111
output bit:	0	0	0	1	0	1	1	1

Lattice:

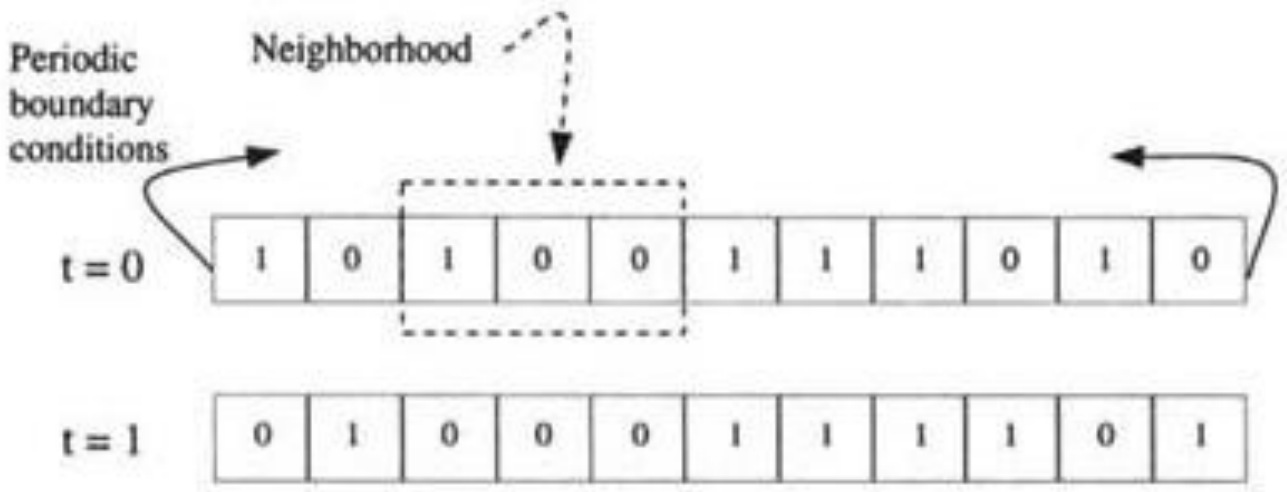
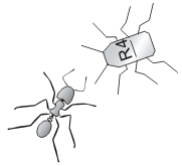


Figura 6: Autómata celular unidimensional binario de largo 11. (Vecindad = 3)

Síntesis de Autómatas Celulares

- Los autómatas celulares presentan arquitecturas de cómputo rápido y confiable y se han estudiado como objetos matemáticos.
- Sin embargo, es difícil diseñar autómatas celulares que tengan un comportamiento esperado. Esta restricción ha limitado su utilización en aplicaciones prácticas.
- Un objetivo deseable es utilizar AE como metodología para obtener las reglas de un autómata celular capaces de llevar a cabo cierta operación compleja en particular.
- Típicamente, el comportamiento de un autómata celular puede ser interpretado como una entrada – la configuración inicial –, una salida – la configuración después de algunos pasos de tiempo – y los pasos de cómputo – los pasos intermedios que transforman la entrada en la salida.



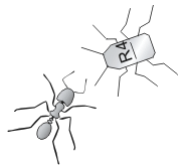
Síntesis de Autómatas Celulares

- El “programa” está constituido por las reglas utilizadas por cada celda del autómata celular.

- Las reglas que se generan sirven para coordinar las decisiones entre las celdas y contribuir a obtener el resultado deseado.

- No existe una unidad central de proceso o una memoria centralizada que dirija la coordinación.

- El comportamiento de los autómatas celulares se ilustra con diagramas “espacio-tiempo” que muestran las configuraciones obtenidas en cada paso de tiempo representado los unos como celdas negras y los ceros como celdas blancas.



Síntesis de Autómatas Celulares

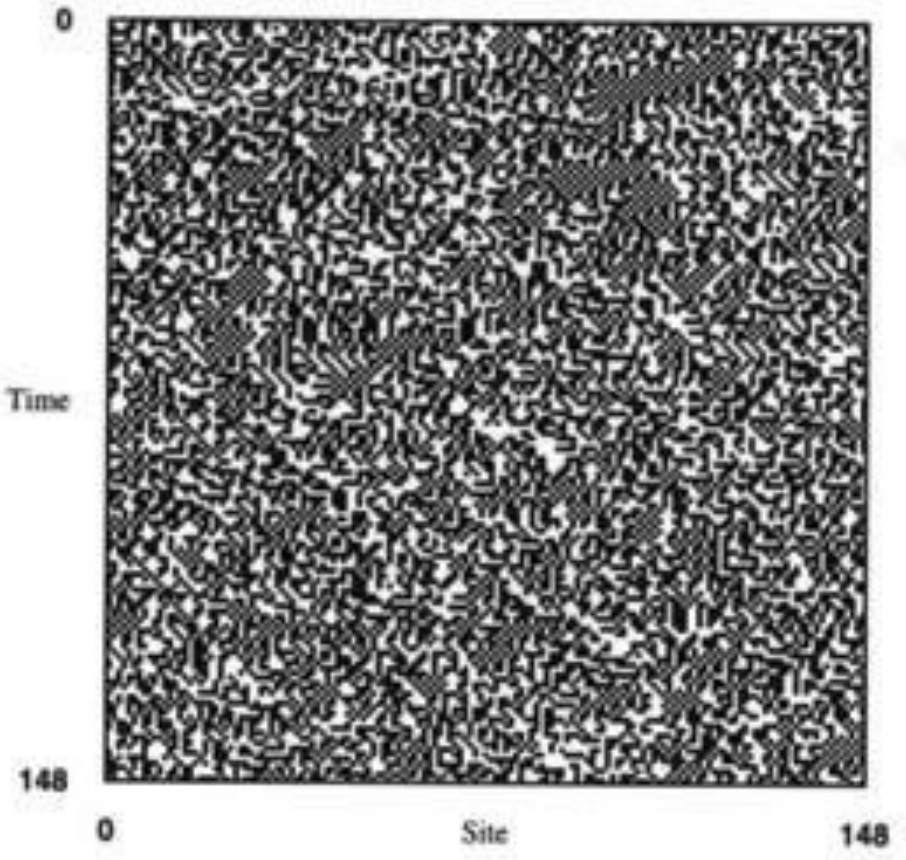


Figura 7: Diagrama "espacio-tiempo" para un CA generado aleatoriamente. (N = 149, vecindad = 7)

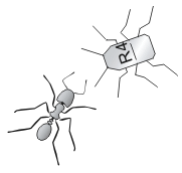
Síntesis de Autómatas Celulares

- A continuación se presenta el trabajo de Mitchell, Hraber, Crutchfield y Das sobre la aplicación de técnicas evolutivas al problema de sintetizar un programa para autómatas celulares, con un comportamiento deseado a priori.

- Estos trabajos datan de 1993 y 1994 y son continuadores de los trabajos realizado por Packard en 1988.

- El objetivo consiste en utilizar autómatas para realizar tareas de clasificación de densidad, que decidan si la configuración inicial contiene mayoría de unos o mayoría de ceros.

- Si tiene mayoría de unos debe terminar con una configuración de todos unos, en otro caso de todos ceros.

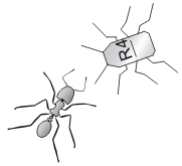


Síntesis de Autómatas Celulares

- Sea ρ_0 la densidad de unos en la configuración inicial.
  - Si $\rho_0 > ½$, en M pasos se debe llegar a la configuración de todos unos.
  - En otro caso, en M pasos se debe llegar a la configuración de todos ceros.
  - M es un parámetro que depende del largo del lattice N.

- La regla de la mayoría, ¿ no cumple con lo solicitado ?

- En la siguiente figura se utilizan vecindades de 7 celdas para dos configuraciones iniciales.



Síntesis de Autómatas Celulares

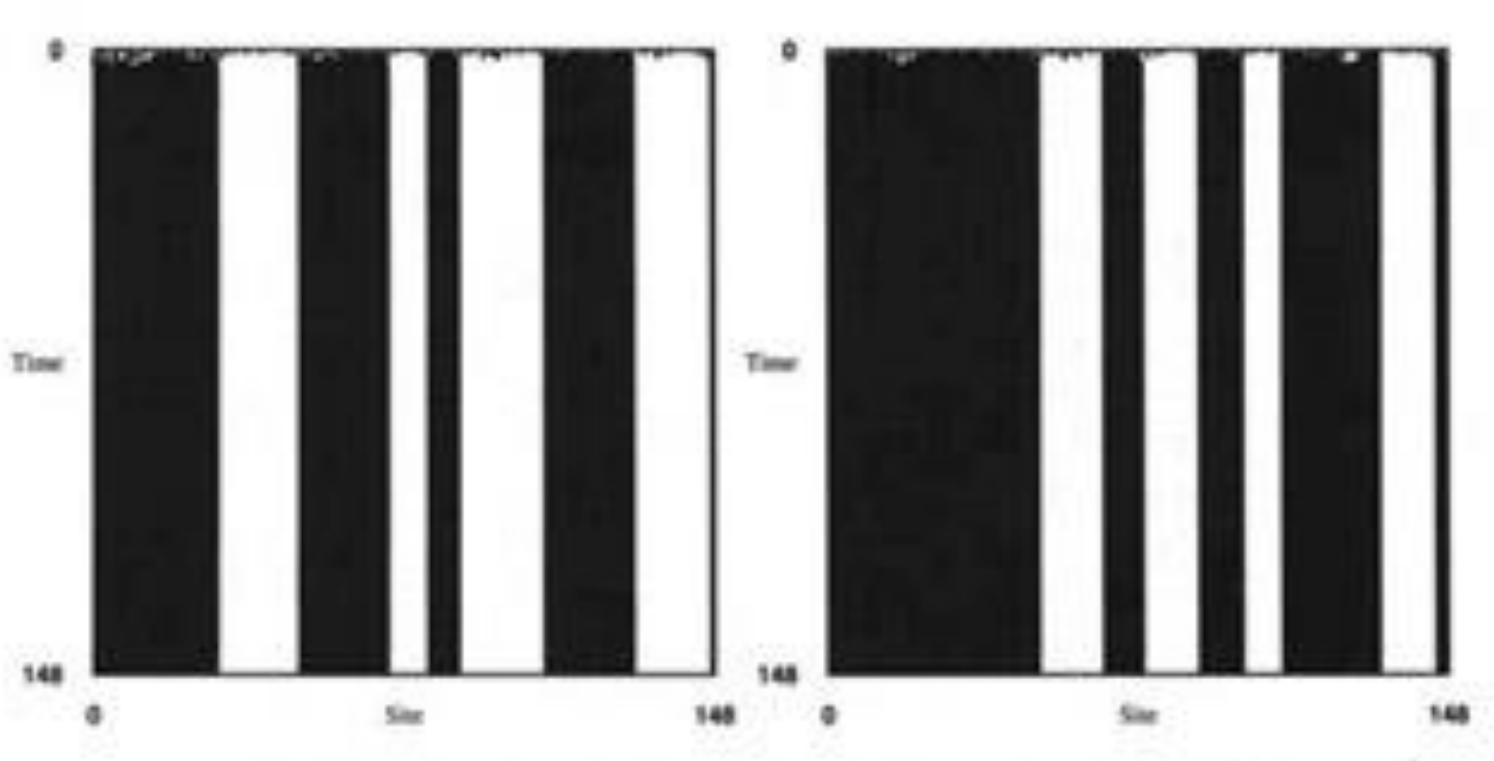


Figura 8: Diagrama "espacio-tiempo" para la regla de la mayoría con vecidad 7.

A la izquierda $\rho_0 < ½$, un CA a la derecha $\rho_0 > ½$.

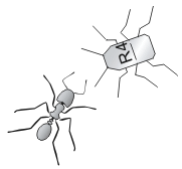
Síntesis de Autómatas Celulares

- Parece funcionar bien en ciertas regiones que tienen mayorías de unos o de ceros, pero cuando se encuentran regiones de todos unos y de todos ceros, no se logra tomar una decisión y ambas regiones se mantienen.

- La dificultad del problema radica en que los unos pueden estar distribuidos a lo largo del lattice, con lo cual la información debe ser transmitida sobre distancias largas.

- Se requiere coordinación global entre celdas que están separadas y que no pueden comunicarse directamente.

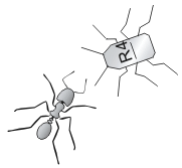
- La propuesta consiste en utilizar un AE para descubrir los mecanismos de coordinación.



AE para Síntesis de Autómatas Celulares

Representación

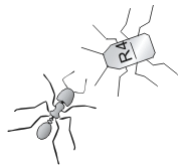
- La representación utilizada es la secuencia de bits que representa la tabla de reglas del autómata celular .
- Cada cromosoma contiene el bit de salida de una regla de la tabla, y está ordenado lexicográficamente por la vecindad.
- Se consideran vecindades de largo 7, por lo cual el largo del cromosoma es 128 (2^7).
- El espacio de búsqueda tiene dimensión 2^{128}.
- El largo del lattice es N = 149.



AE para Síntesis de Autómatas Celulares

Función de fitness

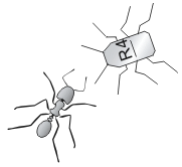
- Para calcular el fitness de una tabla de reglas:
  - Se generan 100 configuraciones iniciales aleatorias de $\rho \in [0,1]$ (dist. uniforme), con exactamente la mitad < ½ y la mitad > ½.
  - Se ejecutan las reglas para cada configuración inicial hasta que no se produzcan modificaciones o hasta un máximo de 2N pasos.
  - Se determina si el resultado obtenido es correcto.
- El fitness es f_{100}, la fracción de las 100 configuraciones iniciales en cuales la tabla de reglas obtuvo el resultado correcto.
- Solamente se consideran como correctos los que llegan exactamente a la configuración deseada.



AE para Síntesis de Autómatas Celulares

Funcionamiento del AG

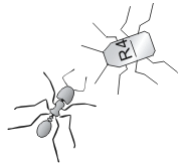
- Se utilizó una población de 100 individuos, inicializados aleatoriamente.
- El proceso evolutivo se repite durante 100 generaciones
  - Se generan 100 nuevas configuraciones iniciales aleatorias.
  - Se calcula f100 para cada tabla de reglas de la población.
  - La población se ordena según su fitness.
  - El mejor 20% de las tablas de reglas se copia sin cambios a la próxima generación.
  - El resto se obtiene mediante el cruzamiento de un punto entre pares de individuos seleccionados aleatoriamente del 20% mejor.
  - Se aplica mutación dos veces sobre cada individuo generado por el cruzamiento.



AE para Síntesis de Autómatas Celulares

- El método de selección presenta similaridades con el utilizado por las estrategias de evolución ($\mu+\lambda$).
- En la Figura 9 se presenta el comportamiento de algunas reglas que fueron derivadas por el algoritmo genético.
- A modo de ejemplo, la regla φ_A, podría expresarse como:

  "*Llevar la configuración a todos ceros, a menos que haya bloques suficientemente grandes de unos adyacentes o casi adyacentes. En ese caso, expandir el bloque de unos*."



AE para Síntesis de Autómatas Celulares

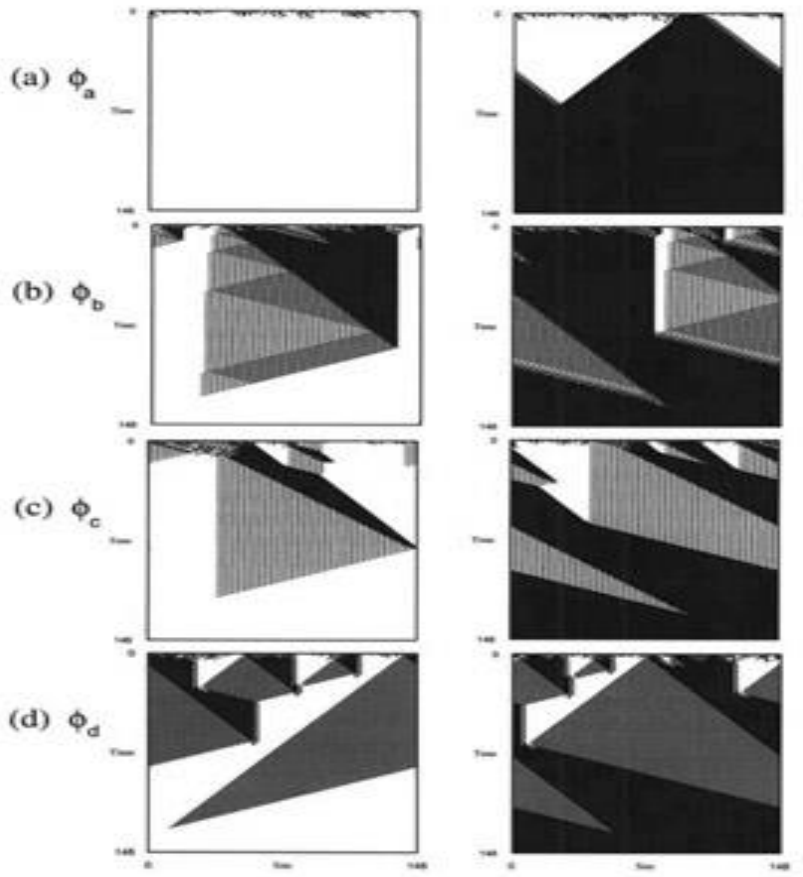
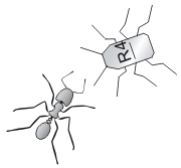


Figura 9: Diagrama "espacio-tiempo" para cuatro reglas diferentes descubiertas por el AG. A la izquierda $\rho_0 < ½$, un CA a la derecha $\rho_0 > ½$.

Algoritmos Evolutivos Paralelos

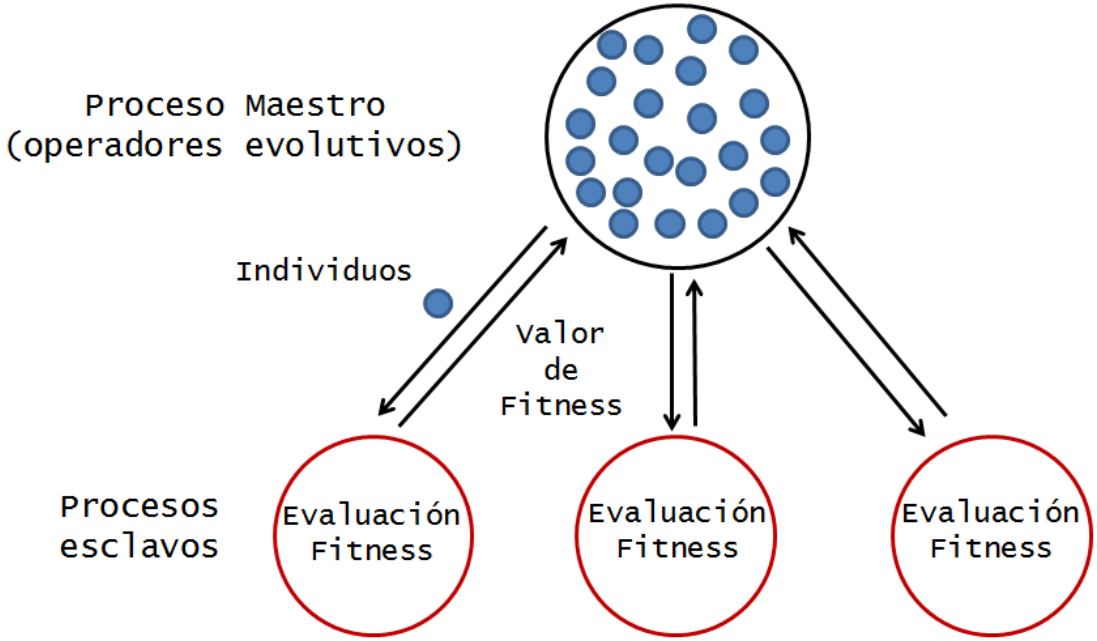
- Las técnicas de paralelismo se aplican a los AE con los siguientes objetivos:
  - Mejorar la eficiencia.
  - Perfeccionar la calidad del mecanismo de búsqueda genética.

- El principal criterio de clasificación surge de las diferentes formas de organización de la población.



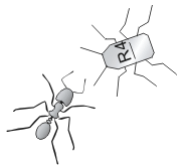
Modelo Maestro-Esclavo

- Las unidades de cómputo procesan distintas etapas del AE, en general: la evaluación de la función de fitness, ya que involucra un tiempo de ejecución mayor que el de los operadores evolutivos.
- Interacción panmíctica: cada individuo tiene la posibilidad de interactuar, competir o cruzarse con cualquier otro (como secuencial)



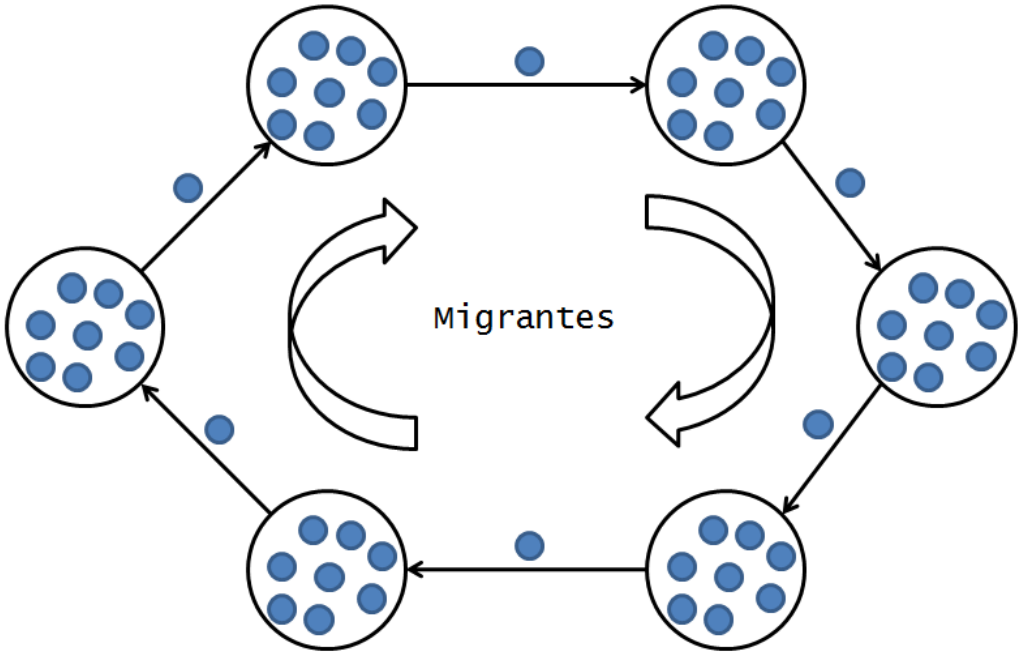
Algoritmos Evolutivos Paralelos

- Los AE paralelos no tienen el mismo funcionamiento que un AE secuencial (excepto el modelo maestro-esclavo).
- Los mecanismo de interacción en el resto de los AEP son distintos a panmíctica.
- Esas distintas interacciones pueden provocar mejoras en el mecanismo de búsqueda del algoritmo.



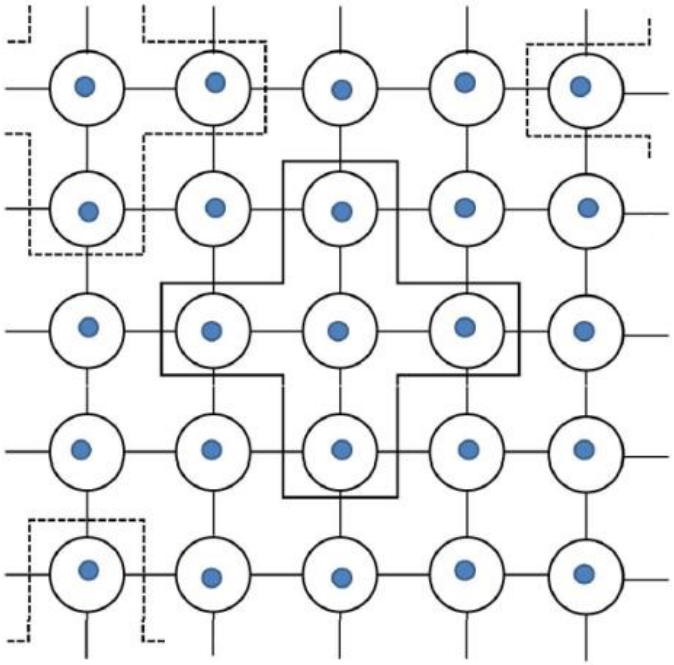
Modelo distribuido o islas

- Se distribuye la población en subpoblaciones (islas), que evolucionan de modo semi-independiente.
- Un operador de migración comunica individuos entre islas.



Modelo celular o AE Celular

- La población se estructura en muchos vecindarios solapados.
- Cada individuo se coloca en una celda en una cuadrícula n-dimensional toroidal y pertenece a varios vecindarios.
- La interacción entre individuos es limitada: la selección para la reproducción y la reproducción es local en cada vecindario.
- El efecto de encontrar soluciones de alta calidad se extiende gradualmente a lo largo de la red debido al uso de un modelo de difusión que es consecuencia de la superposición de vecindarios.



Multi-objective Cellular Genetic Algorithm

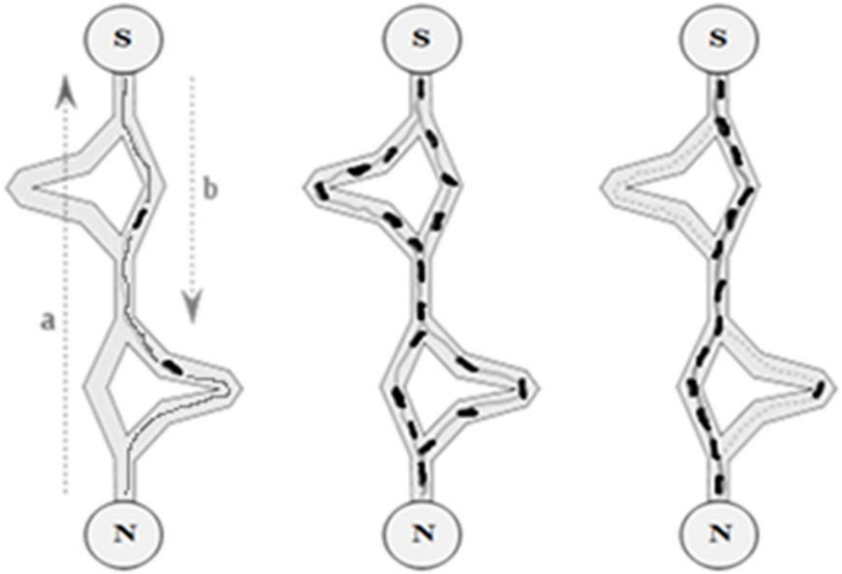
- cGA especialmente diseñado para la optimización multiobjetivo.
- Utiliza un archivo externo para almacenar el conjunto de soluciones no dominadas encontradas por el algoritmo.
- La solución actual se reemplaza por la nueva solución solo si esta la domina.
- Si ambas soluciones son no dominadas, la peor solución del vecindario es reemplazada por la nueva.

```
1  paretoFront = emptySet()
2  popGrid = generateInitialPopulation()
3  evaluateFitness(popGrid)
4  generation = 0
5  insertInFront(paretoFront,popGrid)
6  while not stopCondition() do
7      for individual = 1 to popSize do
8          currentPosition = getPosition(individual)
9          nList = getNeighborhood(popGrid,currentPosition)
10         parent1 = selection(nList)
11         parent2 = selection(paretoFront)
12         offspring = crossover(cp,parent1,parent2)
13         offspring = mutation(mp,offspring)
14         evaluateFitness(offspring)
15         replacement(popGrid,currentPosition,offspring,nList)
16         if individual not dominates offspring then
17             insertInFront(paretoFront,offspring)
18         end
19     end
20     generation = generation + 1
21 end
```

Algorithm 1: Pseudocode of MoCell.

Ant Colony Optimization

- Basadas en el comportamiento de las hormigas.
- Cada hormiga tiene capacidades limitadas, pero en conjunto logran un comportamiento “inteligente”.
- Comunicación indirecta a través del depósito de feromona que sirve de referencia a otras hormigas.



Ant Colony Optimization

- Idea del algoritmo: cada hormiga artificial construye una solución agregando componentes a una solución parcial en consideración.
- La decisión sobre qué componentes agregar es probabilística, a partir de la feromona depositada y un factor heurístico (para asegurar la exploración).

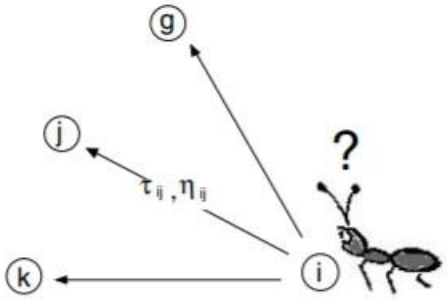


Figure 3.2
An ant arriving in city i chooses the next city to move to as a function of the pheromone values τ_{ij} and of the heuristic values η_{ij} on the arcs connecting city i to the cities j the ant has not visited yet.

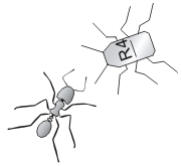
- Al terminar de construir la solución, la regla de actualización incrementa la feromona en las componentes con “mejor calidad”.

Particle Swarm Optimization

- Técnica inspirada en el comportamiento social de bandadas de aves y bancos de peces.
- Si un ave encuentra un camino atractivo, el resto de la bandada puede seguirlo, incluso si están en el lado opuesto de la bandada.

Particle Swarm Optimization

- Se modela con partículas que tienen atributos de posición y velocidad en un espacio multidimensional.
- Las partículas se mueven en ese espacio y recuerdan la mejor posición que conocen.
- En cada paso del algoritmo, las partículas actualizan su velocidad y su posición.
- Cada partícula pertenece a un vecindario.



Particle Swarm Optimization

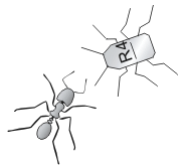
PSO: Velocity update

Basic update rule:

$$\mathbf{v}_i := \mathbf{v}_i + \rho_1 \cdot (\mathbf{p}_i - \mathbf{x}_i) + \rho_2 \cdot (\mathbf{p}_g - \mathbf{x}_i)$$

Consists of:

1. Momentum term: $\mathbf{v}_i$
   → reinforces the previous direction
2. Cognitive part: $\rho_1 \cdot (\mathbf{p}_i - \mathbf{x}_i)$
   → represents the influence of the best solution seen so far by i
3. Social part: $\rho_2 \cdot (\mathbf{p}_g - \mathbf{x}_i)$
   → represents the influence of the best solution seen by the neighborhood of i

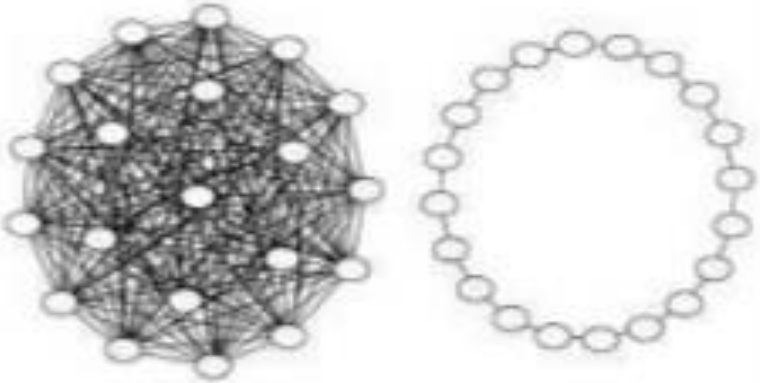


Particle Swarm Optimization

PSO: neighborhoods (1)

Basic division:

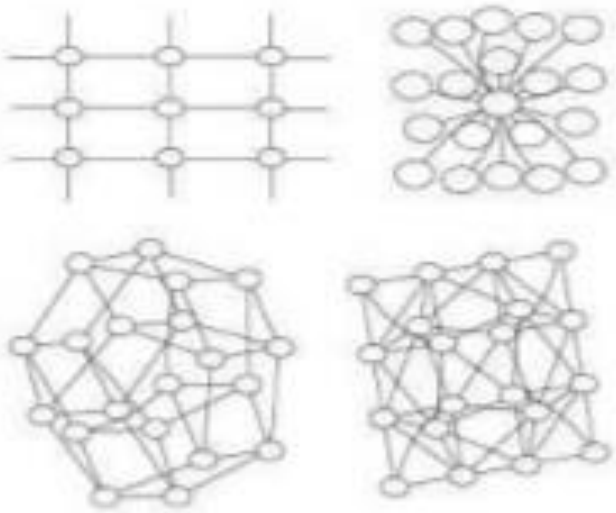
1. gbest PSO: The neighborhood of each particle is the whole swarm
2. lbest PSO: Neighborhoods are restricted



Particle Swarm Optimization

PSO: neighborhoods (2)

More sophisticated neighborhoods:



Examples:

- Von Neumann
- Star
- Pyramid
- Dynamic