MGS
a DSL for modeling and simulating $(DS)^2$

Some demonstrations

Antoine Spicher

www.spatial-computing.org/mgs

SUPMECA

June 2015

Outline

- Lindemayer Systems
- Chemical-like Systems
- Cellular Automata
- Multi-agent Systems

Outline

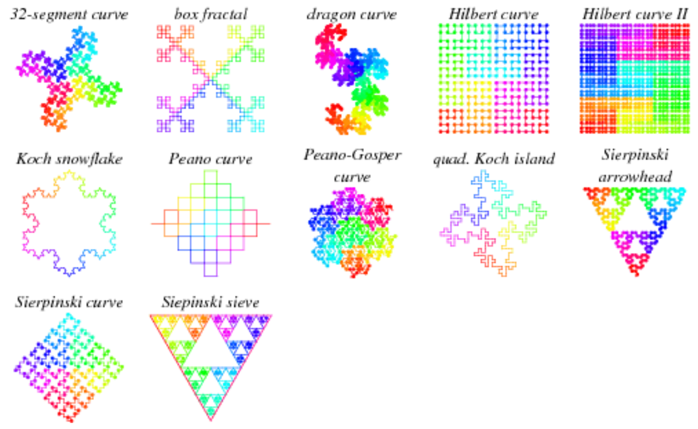
- Lindemayer Systems

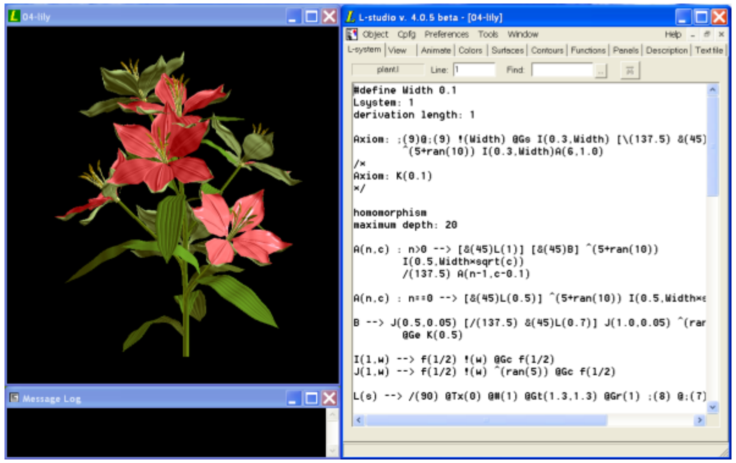
Lindenmayer Systems

- Short description
  - Generative grammar working on sequences of symbols, called *words*
  - Grammar rules $\alpha \to \beta$ where α and β are words + starting axiom ω_0
  - Maximal-parallel application of the rules
    - Rules are applied in parallel everywhere in a word
    - Formally $\omega_i = \omega'_i \; \alpha \; \omega''_i$ becomes $\omega_{i+1} = \omega'_{i+1} \; \beta \; \omega''_{i+1}$
      - If α is found, it is replaced by β
      - ω'_i and ω''_i are transformed independently

Lindenmayer Systems

- Short description
  - Generative grammar working on sequences of symbols, called *words*
  - Grammar rules $\alpha \rightarrow \beta$ where α and β are words + starting axiom ω_0
  - Maximal-parallel application of the rules
    - Rules are applied in parallel everywhere in a word
    - Formally $\omega_i = \omega'_i\ \alpha\ \omega''_i$ becomes $\omega_{i+1} = \omega'_{i+1}\ \beta\ \omega''_{i+1}$
      - If α is found, it is replaced by β
      - ω'_i and ω''_i are transformed independently




04-lily
Message Log
L-studio v. 4.0.5 beta - [04-lily]
Object Cpfg Preferences Tools Window Help
L-system View Animate Colors Surfaces Contours Functions Panels Description Text file
plant.l Line: 1 Find:

```
#define Width 0.1
Lsystem: 1
derivation length: 1

Axiom: ;(9)@;(9) !(Width) @Gs I(0.3,Width) [\(137.5) &(45)
        ^(5+ran(10)) I(0.3,Width)A(6,1.0)
/*
Axiom: K(0.1)
*/

homomorphism
maximum depth: 20

A(n,c) : n>0 --> [&(45)L(1)] [&(45)B] ^(5+ran(10))
        I(0.5,Width*sqrt(c))
        /(137.5) A(n-1,c-0.1)

A(n,c) : n==0 --> [&(45)L(0.5)] ^(5+ran(10)) I(0.5,Width*s

B --> J(0.5,0.05) [/(137.5) &(45)L(0.7)] J(1.0,0.05) ^(ran
        @Ge K(0.5)

I(1,w) --> f(1/2) !(w) @Gc f(1/2)
J(1,w) --> f(1/2) !(w) ^(ran(5)) @Gc f(1/2)

L(s) --> /(90) @Tx(0) @#(1) @Gt(1.3,1.3) @Gr(1) ;(8) @;(7)
```

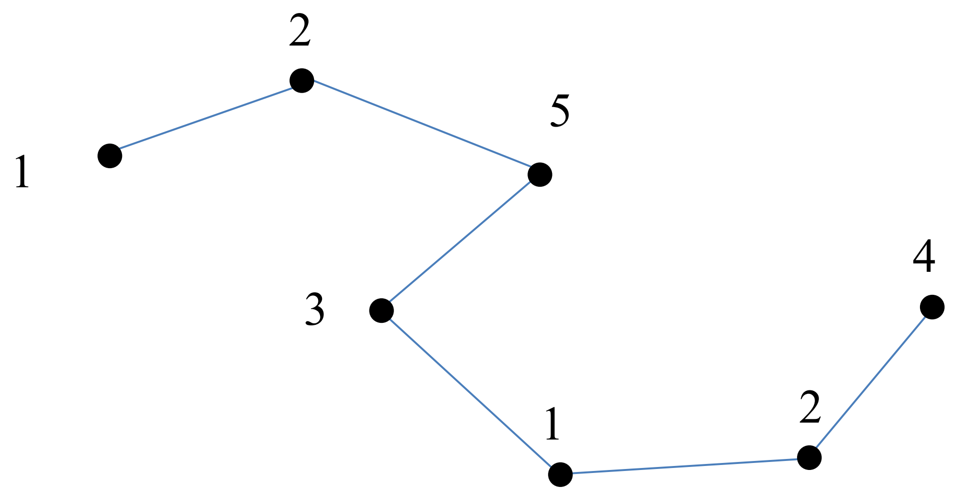
http://http://mathworld.wolfram.com

L-studio, http://algorithmicbotany.org

Lindenmayer Systems

- In MGS
  - Topological collection
    - Words represented by sequence of symbols
      - 0-cells (vertices) labelled by symbols
      - 1-cells (edges) neighborhood (elements accessed one after the other)

```
seq:(1, 2, 5, 3, 1, 2, 4)
```



  - Transformation

    Maximal/parallel rule application strategy (default in MGS)

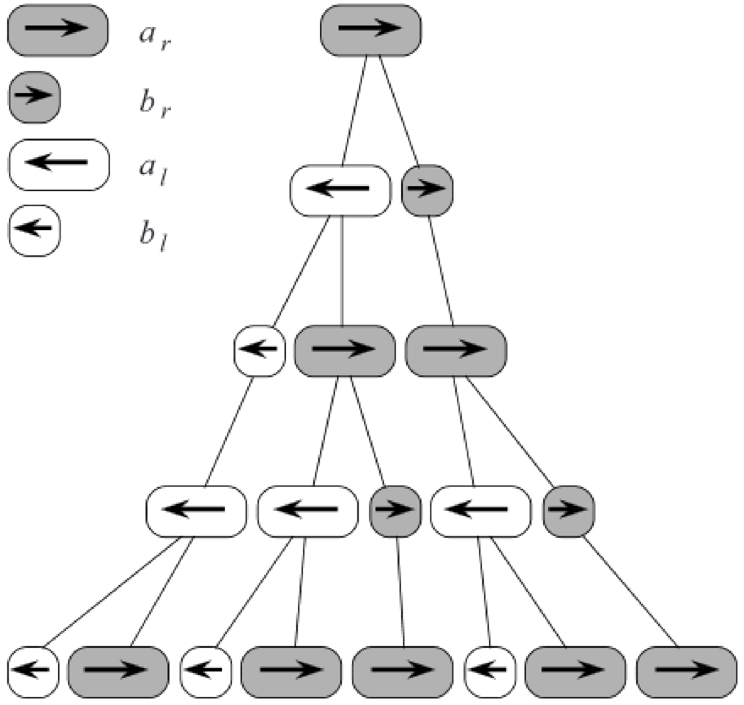
Lindenmayer Systems

- Symbolic growth model of *Anabaena Catenula*
  - Filamentous cyanobacteria

Lindenmayer Systems

- Symbolic growth model of *Anabaena Catenula*
  - Filamentous cyanobacteria
  - Asymmetric division: one daughter is smaller than the other
  - Polarized cell (left/right orientation)

$$\begin{cases} \omega_0 = a_r \\ a_r \rightarrow a_l b_r \\ a_l \rightarrow b_l a_r \\ b_r \rightarrow a_r \\ b_l \rightarrow a_l \end{cases}$$



The Algorithmic Beauty of Plants

Lindenmayer Systems

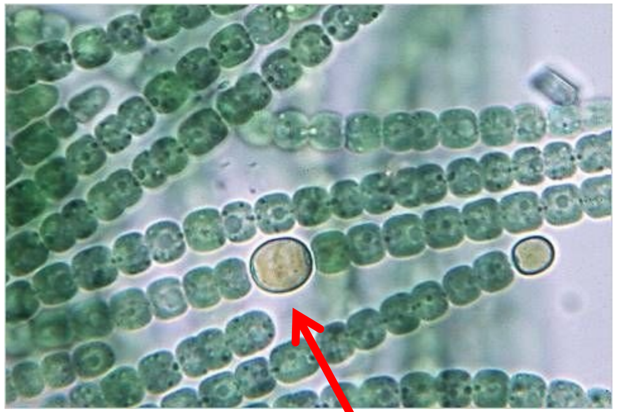
- Symbolic growth model of *Anabaena Catenula*

```
type cell = `Left_Long   | `Right_Long
          | `Left_Short | `Right_Short ;;
type anabaena = [cell]seq ;;

trans grammar = {

  `Right_Short => `Right_Long;

  `Left_Short  => `Left_Long;

  `Right_Long  => `Left_Long, `Right_Short;

  `Left_Long   => `Left_Short, `Right_Long;

} ;;

grammar(seq:(`Right_Long)) ;;
```

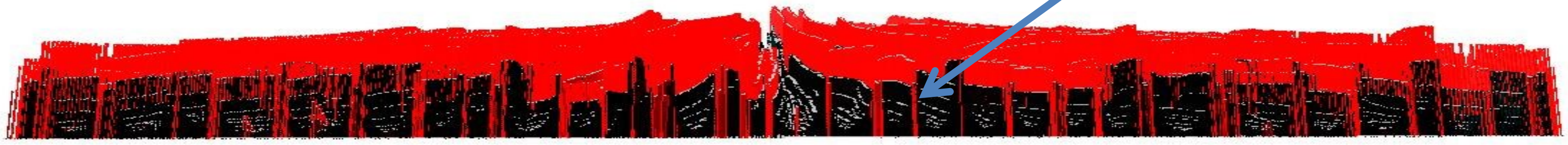
Lindenmayer Systems

- Heterocysts Differentiation in *Anabaena Catenula*
  - Lack of nitrogen
  - Robust structure

    Heterocysts are very regularly distributed (every 10 cells)
  - Wilcox Model
    - Activator/inhibitor
    - Activator triggers the differentiation
    - Activator catalyzes the inhibitor production
    - Inhibitor represses the activator effects (antagonism)
  - L-system implemented in MGS

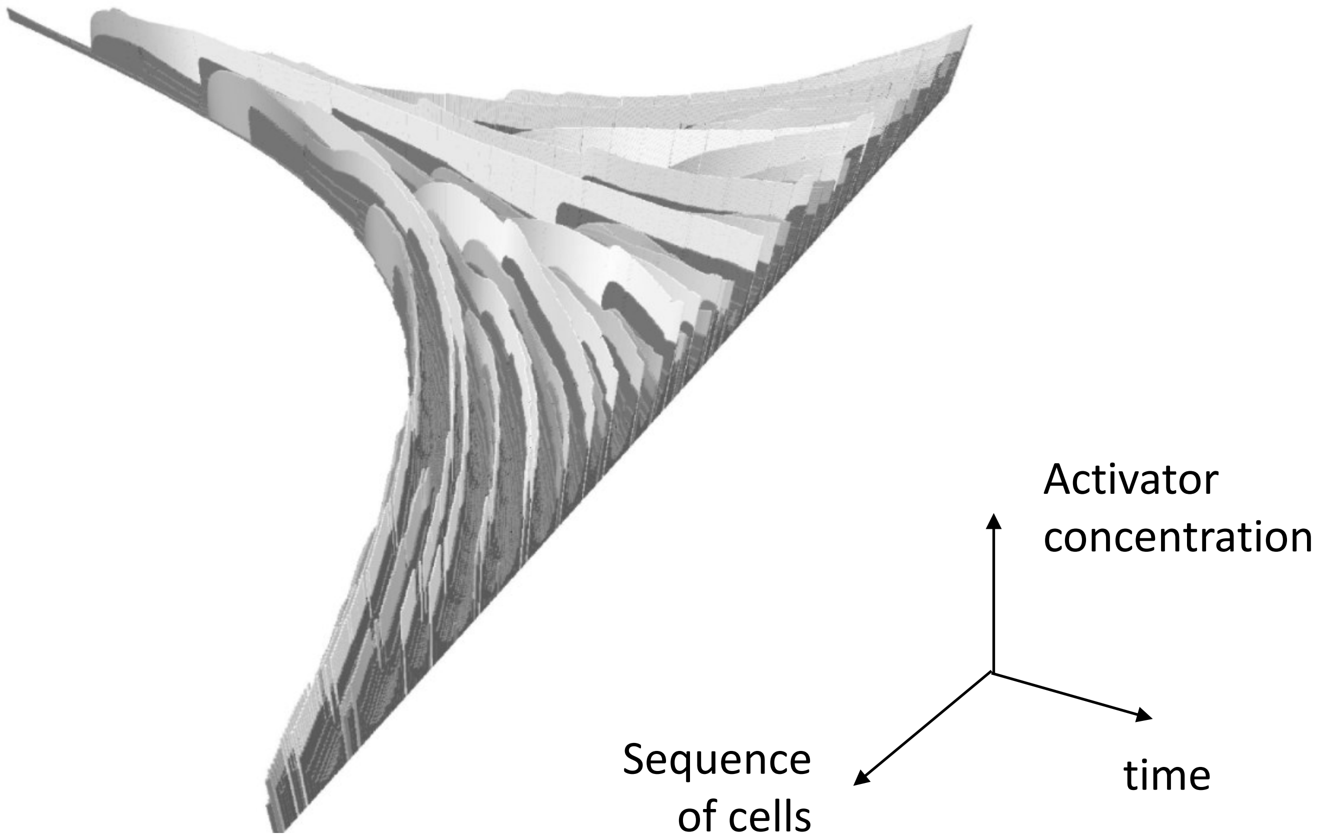


heterocyst



Lindenmayer Systems

- Heterocysts Differentiation in *Anabaena Catenula*



Outline

- Chemical-like Systems

Chemical Modeling

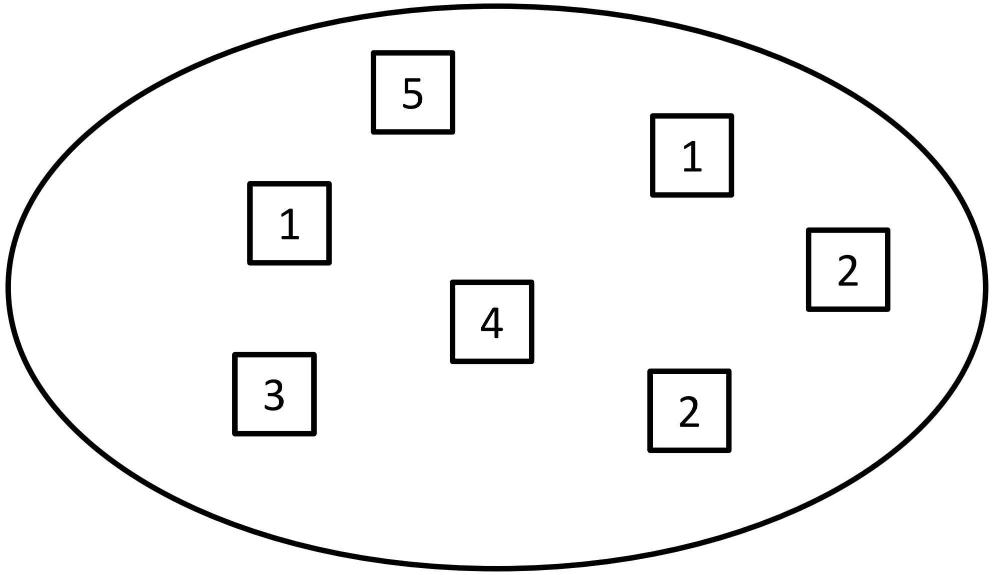
- Short description
  - Model as a chemical system
  - Highly parallel & autonomous
  - *Chemical metaphor*
    - Solution of data (data = chemicals)
    - Dynamics governed by chemical reactions
  - Used in theory of computer science
    - Gamma programming language, Banâtre, Le Metayer, 1986
    - CHAM (CHemical Abstract Machine), Berry, Boudole, 1990
    - Membrane computing

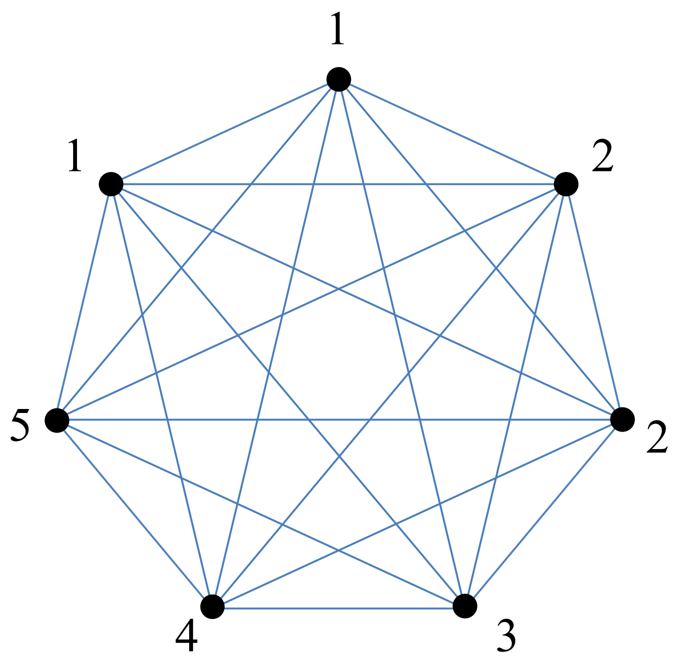
      Extension to nested chemical reactions
  - Can be used for modeling purpose

Chemical Modeling

- In MGS
  - Topological collection
    - Multi-set (bag) of symbols
    - Topology of complete graph

      Any symbol can interact with any other symbol

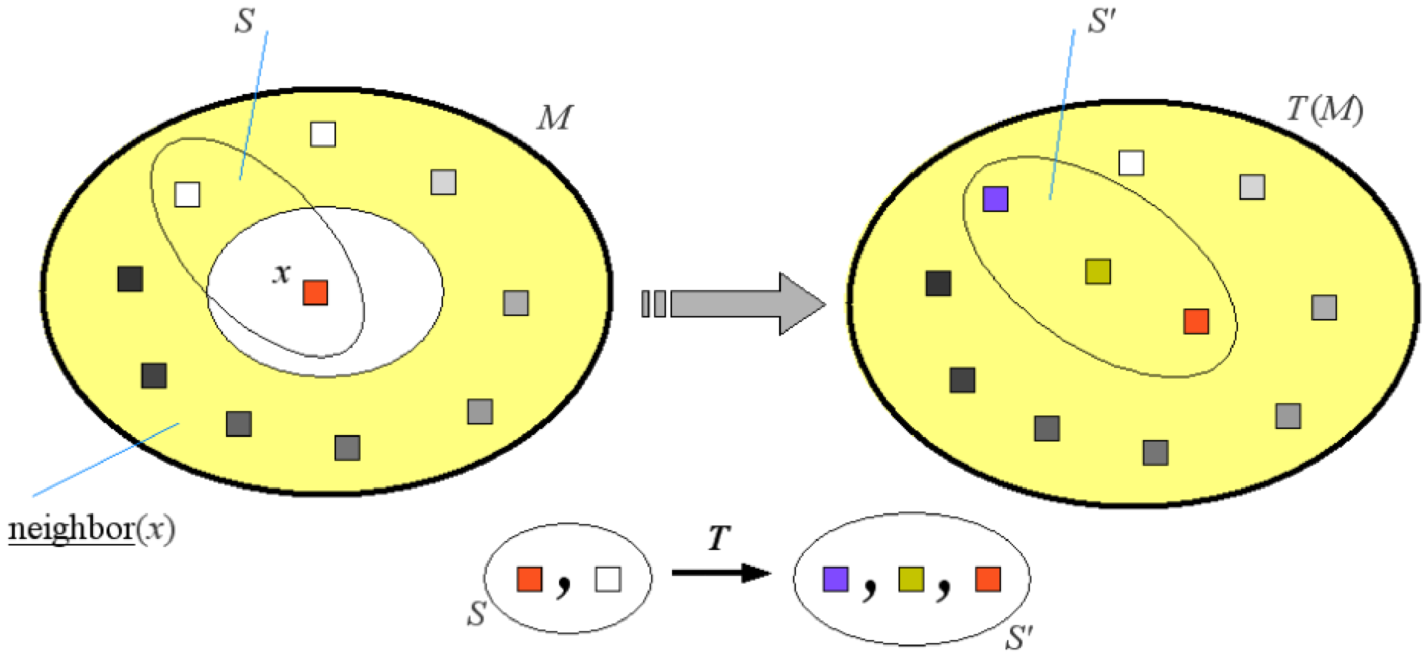




```
bag:(1, 2, 5, 3, 1, 2, 4)
```

Chemical Modeling

- In MGS
  - Transformation T
    - Collection: multi-set M
    - Topology: neighbor$(x) = M \setminus \{x\}$ (any other element)
    - Subcollection: multi-set S



Chemical Modeling

- In MGS
  - Rule application strategies
    - Maximal parallel (used in computing theory)
    - Gillespie's exact Stochastic Simulation Algorithm (1977)
      - Hypothesis

        Data are "well-mixed", only one reaction may occur at a given time
      - Stochastic sequential strategy

        A rule is chosen and applied once w.r.t. some probability law (TCMC)

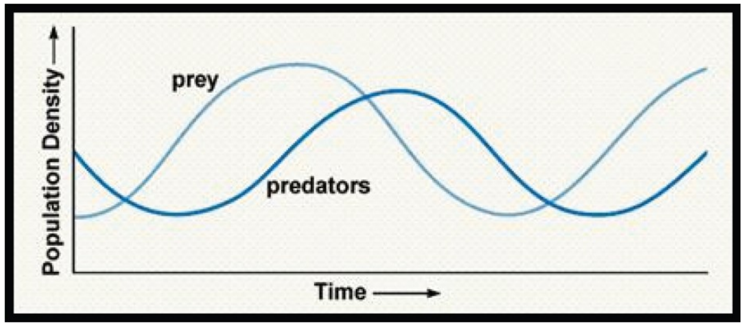
t: current date
τ: elapsed to next reaction
μ: chemical reaction
- c_μ: stochastic constant of reaction μ
- h_μ: number of molecular combinations to activate μ
- $a_\mu = c_\mu h_\mu$: *propensity* of reaction μ

Probability that
- nothing happens in the time interval $(t, t+\tau)$, and
- reaction μ occurs in the time interval $(t+\tau, t+\tau+d\tau)$

$$P(\tau,\mu)d\tau = a_\mu e^{-\tau \sum_\nu a_\nu} d\tau$$

Chemical Modeling

- Lotka-Volterra prey-predator system
  - System exhibiting two interdependent populations, one of which serves as a food source for the other
  - Coupled oscillations
  - Informally
    - Preys spontaneously reproduce
    - Predators spontaneously die
    - Predators hunt preys
      - Preys may die
      - Predators may reproduce
  - Models: ODE and chemical model



Sylvia S Mader, Biology 6th edition, 1998

$$\begin{cases} \dfrac{dV}{dt} = V(\alpha - \beta P) \\ \dfrac{dP}{dt} = P(\gamma V - \delta) \end{cases}$$

$$\begin{cases} V & \xrightarrow{a} & 2\,V \\ V + P & \xrightarrow{b} & P \\ V + P & \xrightarrow{c} & 2\,P \\ P & \xrightarrow{d} & . \end{cases}$$

Chemical Modeling

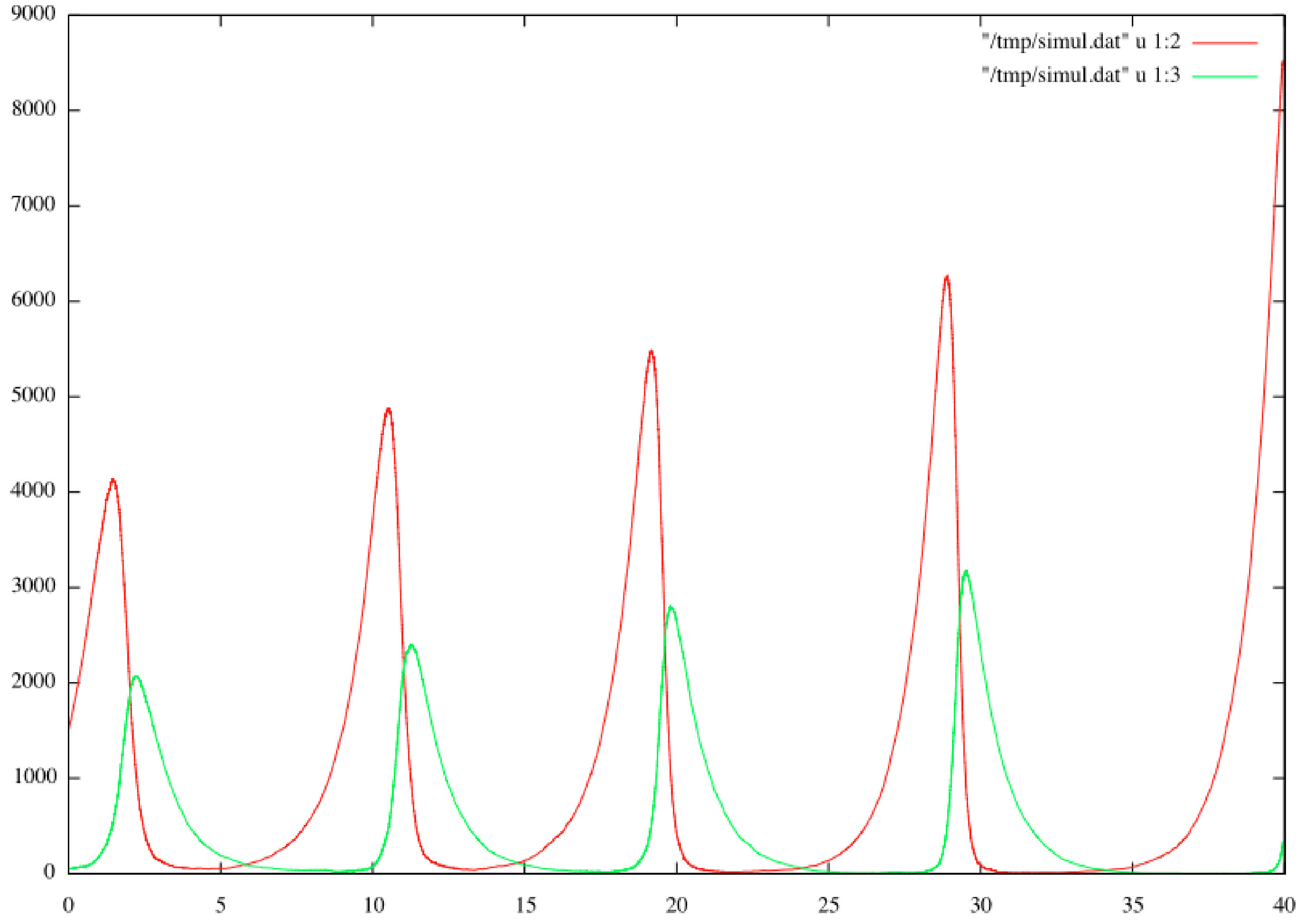
- Lotka-Volterra prey-predator system

```
type chemical = `Prey | `Predator ;;
type population = [chemical]bag ;;

trans reactions = {

   `Prey             ={ C = 1.0   }=> `Prey, `Prey;

   `Predator         ={ C = 1.0   }=> <undef>;

   `Predator, `Prey ={ C = 0.001 }=> `Predator;

   `Predator, `Prey ={ C = 0.001 }=> `Predator, `Predator;

} ;;

reactions[strategy = `gillespie](…) ;;
```

Constant stochastic

Chemical Modeling

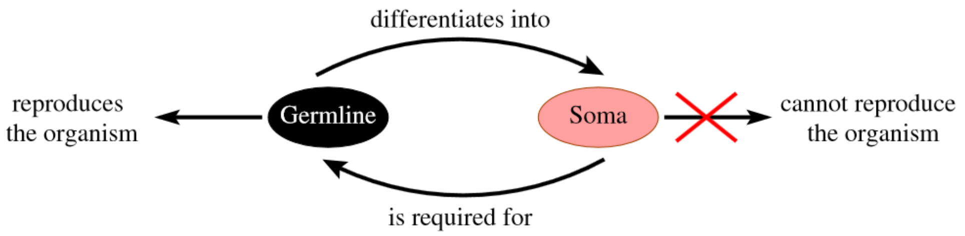
- Lotka-Volterra prey-predator system



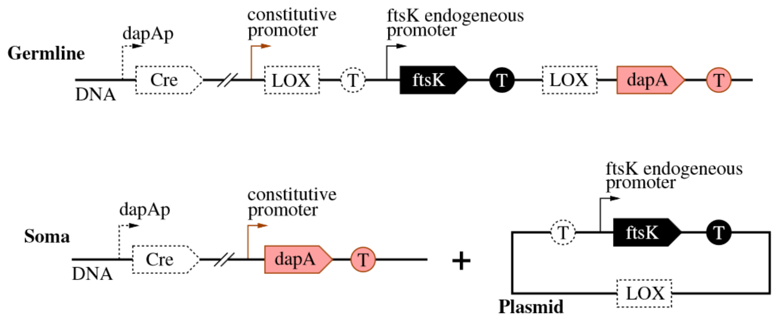
Chemical Modeling

- Synthetic Multi-cellular Bacterium (iGEM project of Paris team in 2007)

Bacterium line able to express a lethal gene without disturbing its growth



Paris team proposal…



… its genetic implementation …

$$\mathrm{Cre} \xrightarrow{C_0} . \qquad \mathrm{D_G} + \mathrm{Cre} \xrightarrow{2C_5} \mathrm{D_G^*}$$

$$\mathrm{DAP} \xrightarrow{C_1} . \qquad \mathrm{D_G^*} + \mathrm{Cre} \xrightarrow{C_5} \mathrm{D_S}$$

$$\mathrm{D_{Cre}} + \mathrm{DAP} \underset{C_{-2}}{\overset{C_2}{\rightleftharpoons}} \mathrm{D_{Cre}^*} \qquad \mathrm{D_S} \xrightarrow{C_6} \mathrm{D_S} + \mathrm{DAP}$$

$$\mathrm{D_{Cre}} \xrightarrow{C_3} \mathrm{D_{Cre}} + \mathrm{Cre} \qquad \mathrm{DAP} \underset{C_{im}}{\overset{C_{ex}}{\rightleftharpoons}} \mathrm{DAP}_{ex}$$

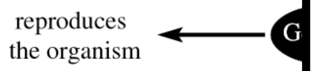
$$\mathrm{D_{Cre}^*} \xrightarrow{C_4} \mathrm{D_{Cre}^*} + \mathrm{Cre}$$

… and its **chemical model**

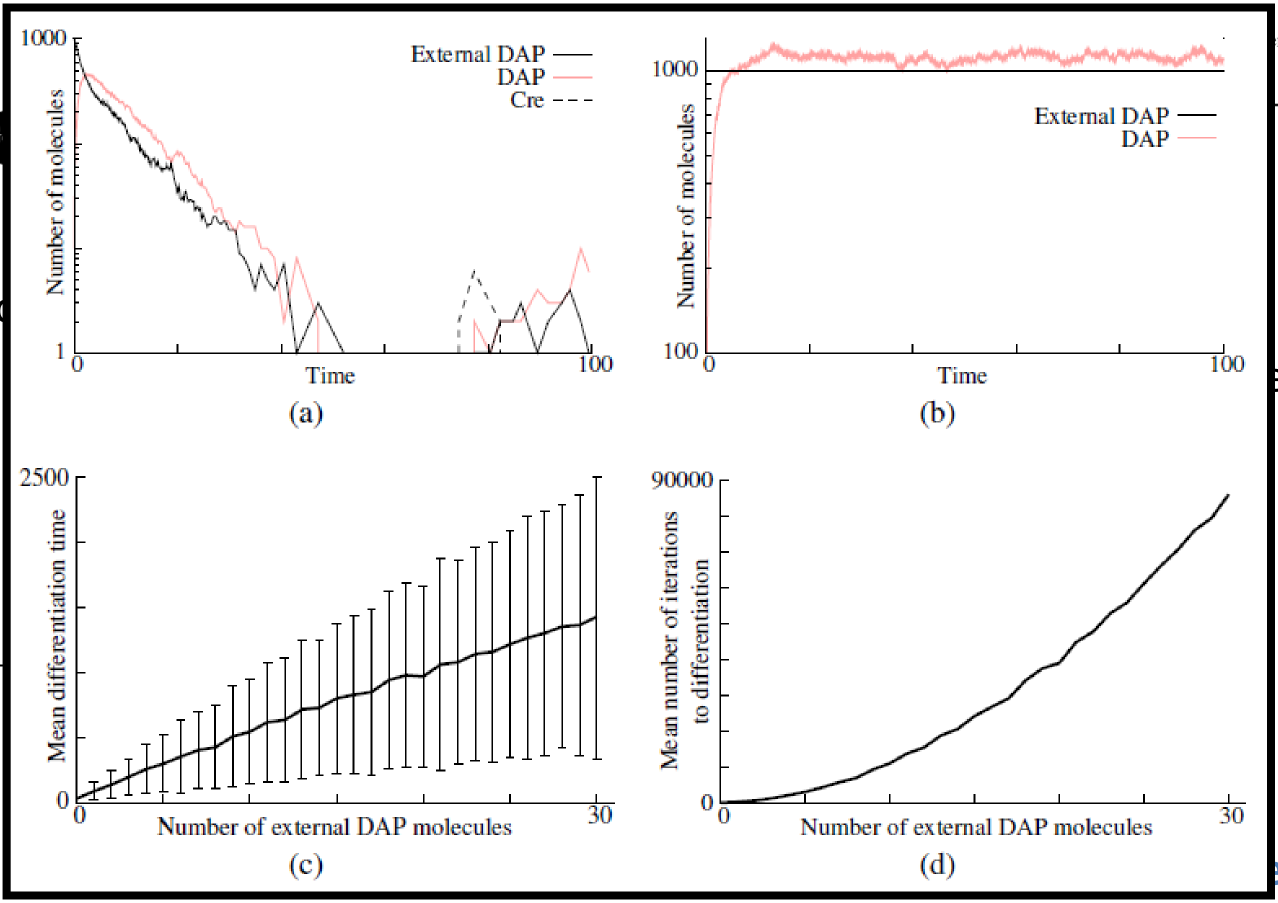
Chemical Modeling

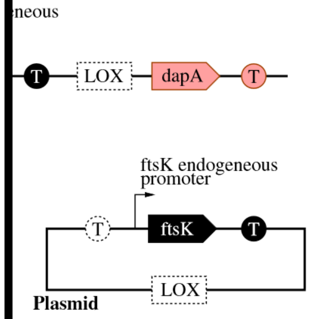
- Synthetic Multi-cellular Bacterium (iGEM project of Paris team in 2007)

Bacterium line able to express a lethal gene without disturbing its growth



Paris team pr





ementation …

D_{Cre} +

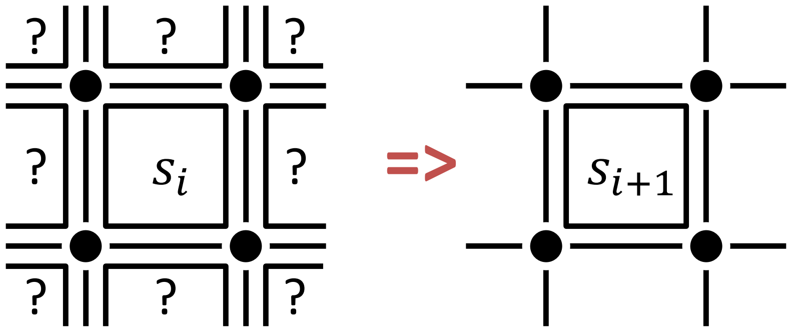
el

Outline

- Cellular Automata

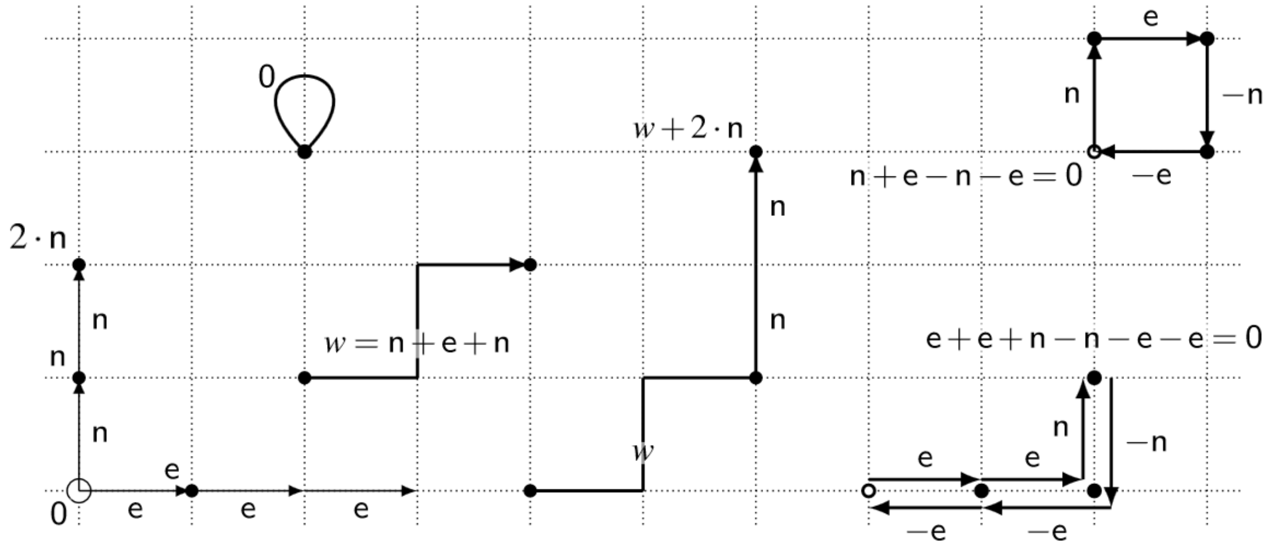
Cellular Automata

- Short description
  - Dynamical systems discrete in space and time
  - Space
    - Set (finite or infinite) of *cells* homogeneously and regularly organized
    - Each cell characterized by its *state*
  - Time
    - Transition function from a *configuration* to another
    - Synchronous update (all cells update their state at the same time)
    - Local specification (as function of the neighbor cells state)



Cellular Automata

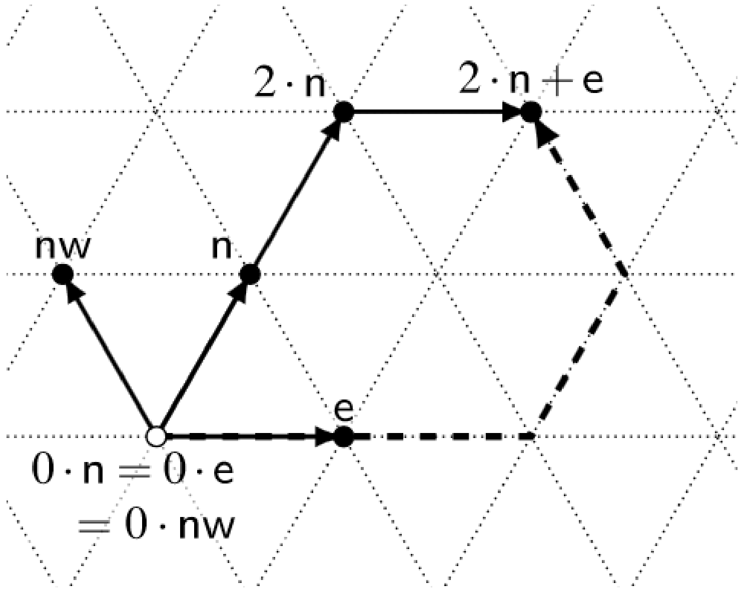
- In MGS
  - Topological collection
    - *Group Based Field* (GBF)
    - Cayley graph associated with a (abelian) group presentation
      - Generators: atomic displacement
      - Relators: displacement properties



```
gbf NEWS = < e, n; e + n = n + e >
```

Cellular Automata

- In MGS
  - Topological collection
    - *Group Based Field* (GBF)
    - Cayley graph associated with a (abelian) group presentation
      - Generators: atomic displacement
      - Relators: displacement properties



```
gbf hexa = < n, e, nw; n = e + nw >
```

Cellular Automata

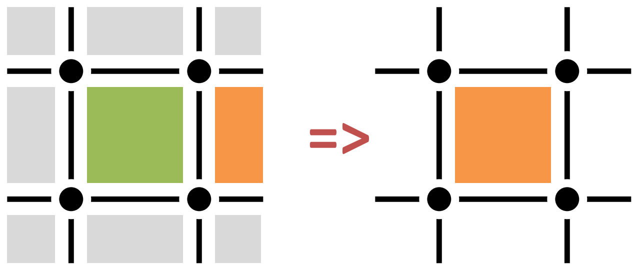
- 3-State fire spread model

🟩 forest

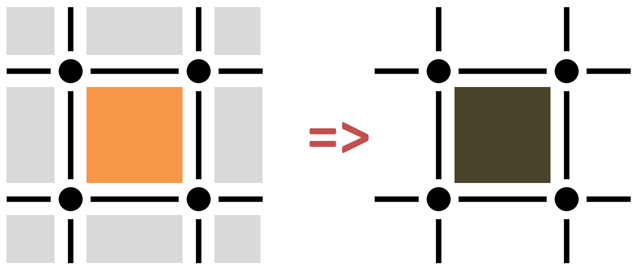
🟧 fire

🟫 ash

⬜ generic



propagation

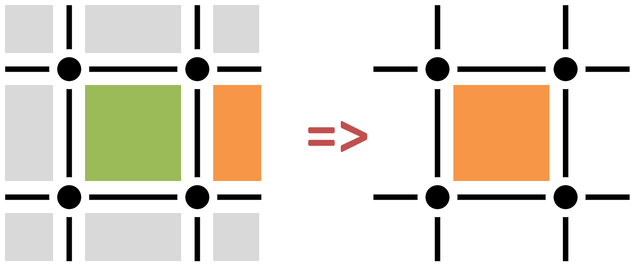


extinction

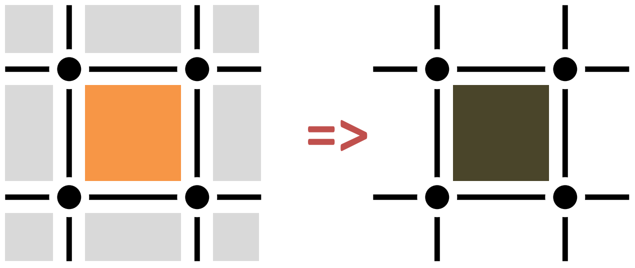
Cellular Automata

- 3-State fire spread model

🟩	forest
🟧	fire
⬛	ash
⬜	generic



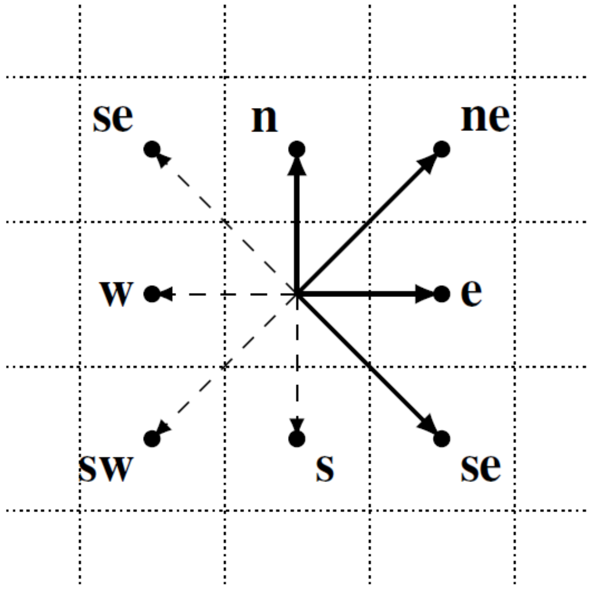
propagation



extinction

```
gbf Moore = < N, NE, E, SE;
              N + E = NE,
              E - N = SE
            > ;;
```

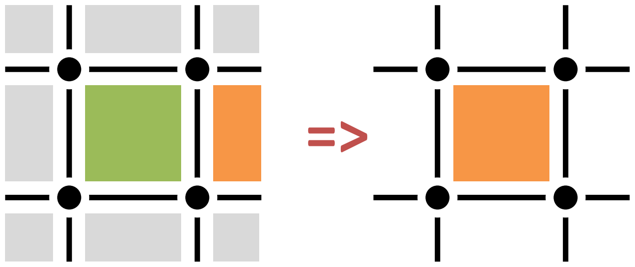
Moore neighborhood, predefined in MGS



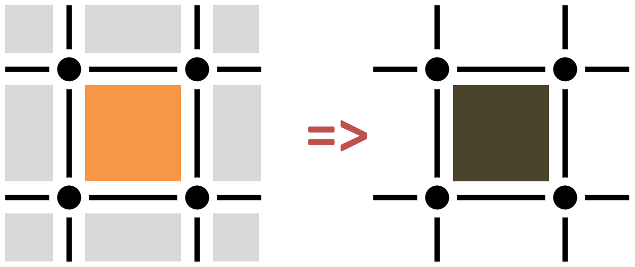
Cellular Automata

- 3-State fire spread model

🟩	forest
🟧	fire
⬛	ash
⬜	generic



propagation



extinction

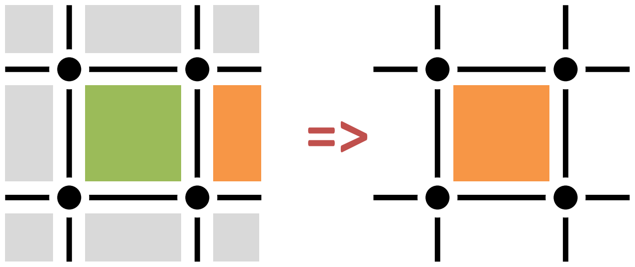
```
type cell = `Forest | `Fire | `Ash ;;
type configuration = [cell]Moore ;;

trans rules = {

  `Forest as c / neighbors_exists(equal(`Fire), c) => `Fire;

  `Fire => `Ash;

} ;;
```

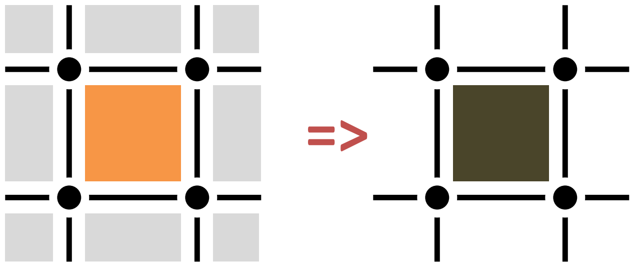
Iterator over the neighbors of c

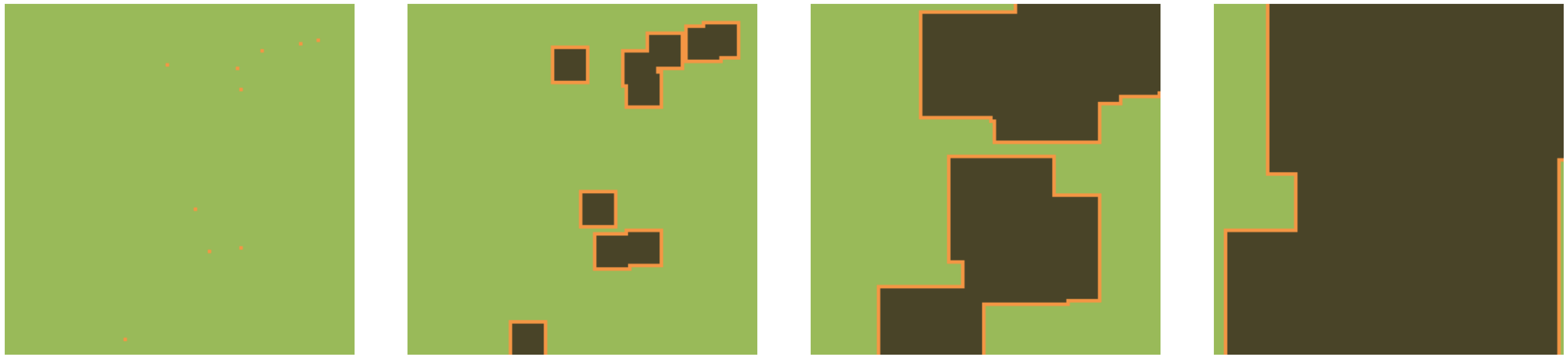
Cellular Automata

- 3-State fire spread model

🟩 forest
🟧 fire
⬛ ash
⬜ generic

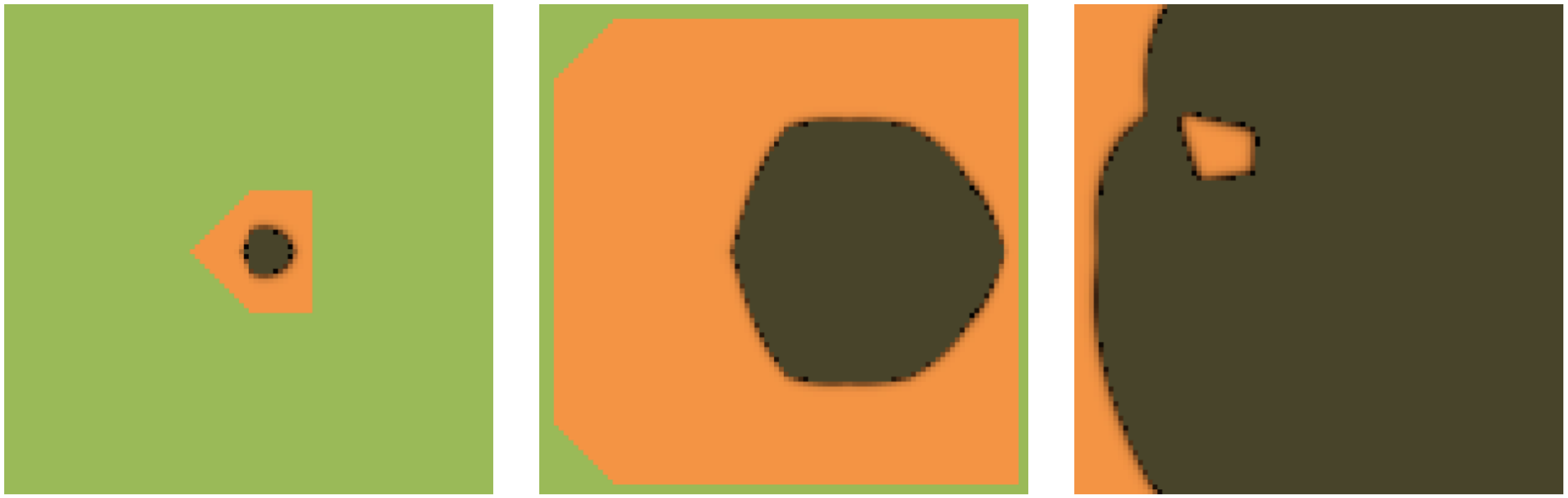

propagation


extinction



Cellular Automata

- Karafyllidis-Thanailakis model
  - More elaborated CA for fire spread
  - Cell state: ratio of burnt area from 0 (none) to 1 (all)
  - Environmental effects
    - Wind (speed and direction)
    - Type of fuel
    - Landscape topography

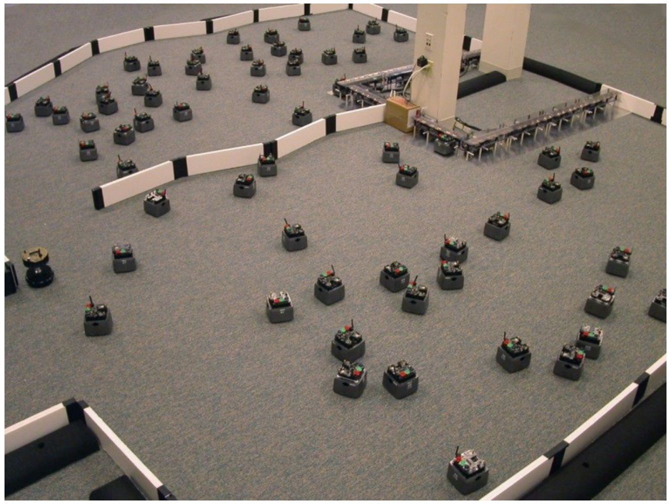


Outline

- Multi-agent Systems

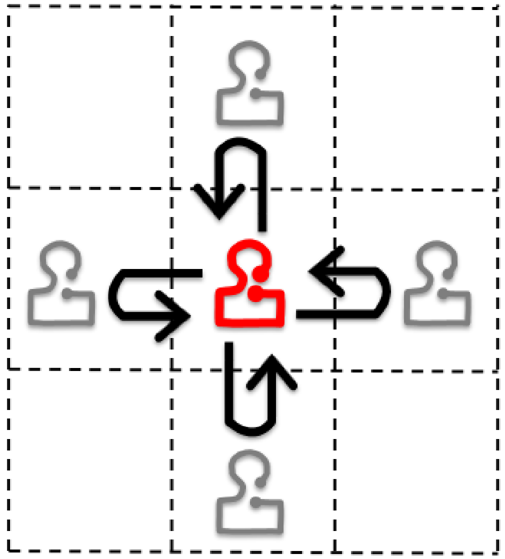
Multi-Agent Systems

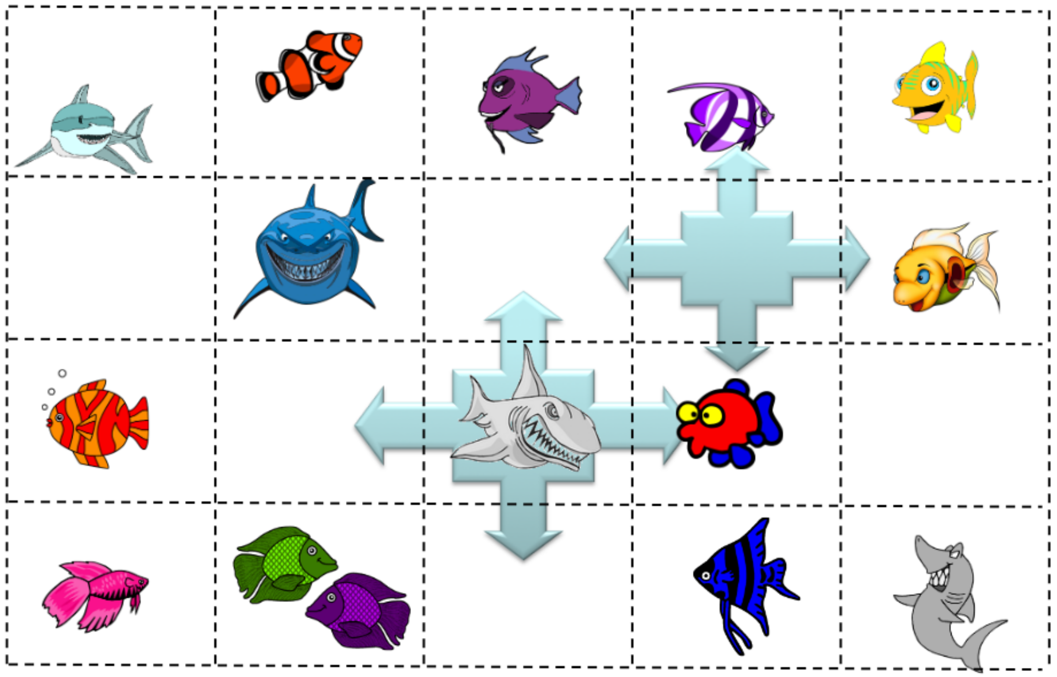
- Short Description
  - Population of entities interacting in some environment
  - *Agents*
    - Characterized by a state
    - Actions
      - Decision procedure
      - Dependence on the nearby environment and neighbors



Multi-Agent Systems

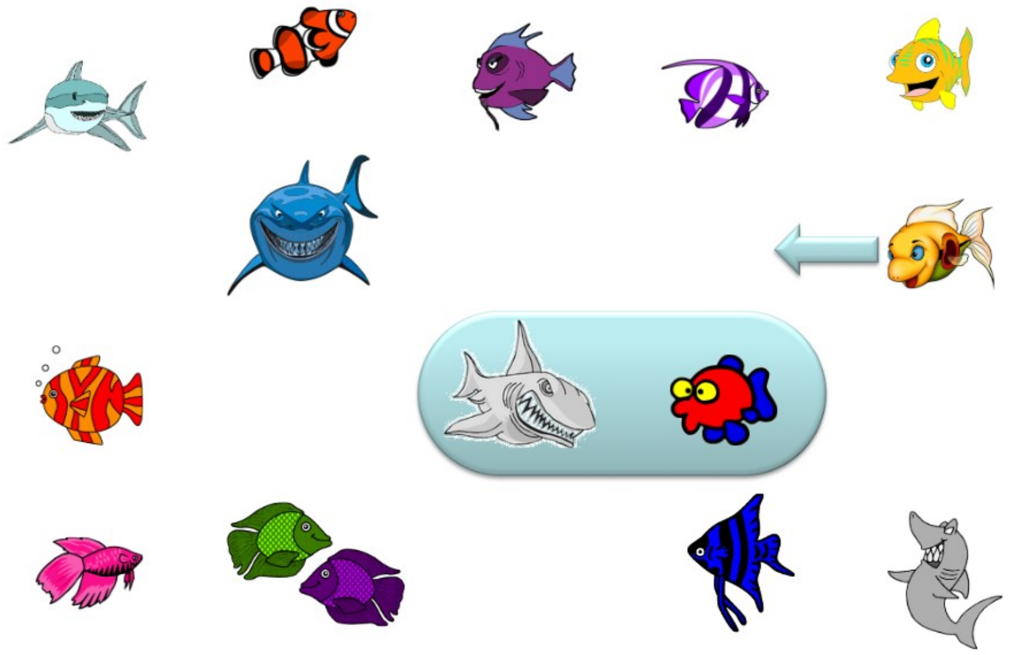
- In MGS
  - Representation of a population of agents
    - Newtonian
      - Structure of the system described through its spatial domain
      - Agents localized in a pre-existing space

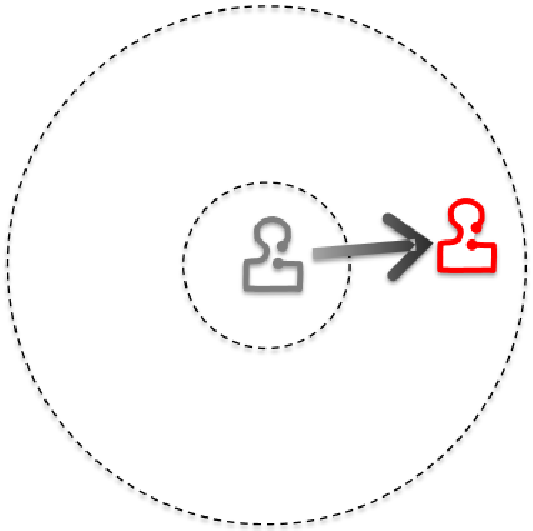




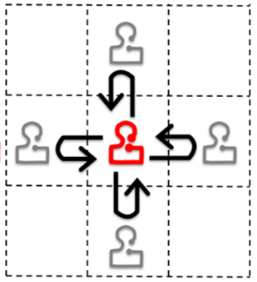
Multi-Agent Systems

- In MGS
  - Representation of a population of agents
    - Leibnizian
      - Structure of the system described through its components
      - Space as a relation between agents



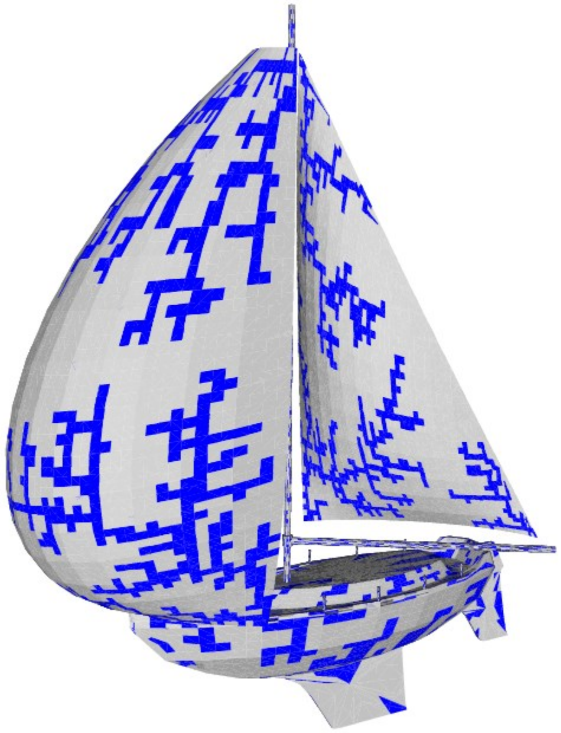


Multi-Agent Systems



- In MGS
  - Representation of a population of agents

    Leibnizian, newtonian
  - Example of a newtonian collection for representing a population



```
type particle = `Mobile | `Fixed ;;
type mas = [particle]Moore ;;

trans behaviors = {

  `Fixed, `Mobile => `Fixed, `Fixed;

  `Mobile, <undef> => <undef>, `Mobile;

} ;;
```

Multi-Agent Systems

- In MGS
  - Representation of a population of agents

    Leibnizian, newtonian
  - Example of a leibnizian collection for representing a population
    - *Geoproximal* topological collection
    - *Two elements are neighbors if they are close enough*

      Agents are embedded in an *n*-dimensional Euclidean space

```
geoprox population(2, 5.0) = fun ag -> (ag.x, ag.y) ;;
```

Dimension of the considered Euclidean space

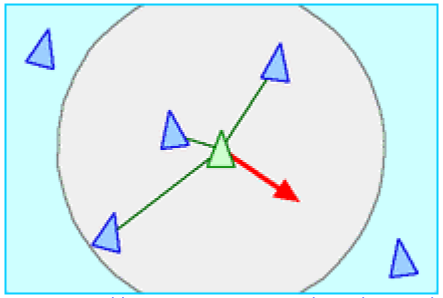
Radius max.

Function to get the coordinates of an agent

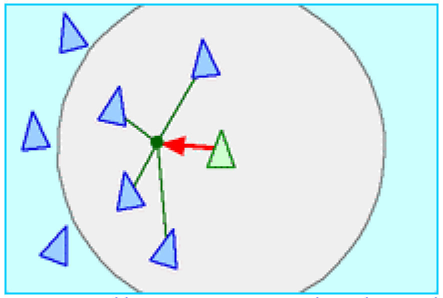
Multi-Agent Systems

- Reynolds' Boids
  - Model explaining flock behaviors of birds, fishes, …

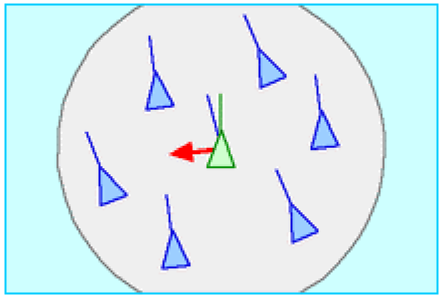
    No leader, simple local behavior rules
  - Agent
    - Virtual bird
    - Positioned and oriented in the 2D space
    - Neighborhood given by a geoproximal with radius 5
  - Three simple behavior rules


http://www.red3d.com/cwr/boids/

Separation: avoid collision with too close mates (dist. < 1)


http://www.red3d.com/cwr/boids/

Cohesion: steer towards neighbors to keep close (dist. > 4)


http://www.red3d.com/cwr/boids/

Alignment: follow the average directions of the mates

Multi-Agent Systems

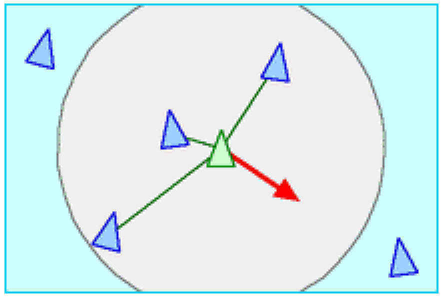
- Reynolds’ Boids

```
record boid = {
  x:float, y:float, t:float
} ;;
geoprox population(2, 5.0) =
  fun b:boid -> (b.x, b.y) ;;
```

Multi-Agent Systems

- Reynolds' Boids

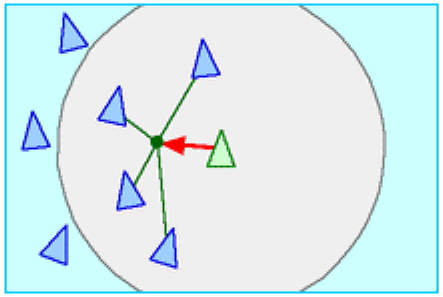
```
record boid = {
  x:float, y:float, t:float
} ;;
geoprox population(2, 5.0) =
  fun b:boid -> (b.x, b.y) ;;

trans behaviors = {  (* see details on the website *)

  b / neighbors_exists(too_close(b), b) => (
    let g = barycenter(b) in
    let dx = b.x - g.x and dy = b.y - g.y  in
    let t = to_angle(dx, dy) in
    let b' = b + { t = t } in
      move_boid(b')
  );
  …
} ;;
```



Multi-Agent Systems

- Reynolds' Boids

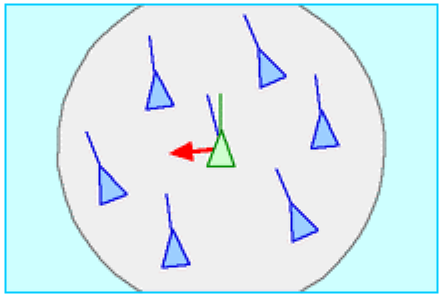
```
record boid = {
  x:float, y:float, t:float
} ;;
geoprox population(2, 5.0) =
  fun b:boid -> (b.x, b.y) ;;

trans behaviors = {  (* see details on the website *)
  …
  b / neighbors_forall(too_far(b), b) => (
    let g = barycenter(b) in
    let dx = g.x - b.x and dy = g.y - b.y  in
    let t = to_angle(dx, dy) in
    let b' = b + { t = t } in
      move_boid(b')
  );
  …
} ;;
```



Multi-Agent Systems

- Reynolds’ Boids

```
record boid = {
  x:float, y:float, t:float
} ;;
geoprox population(2, 5.0) =
  fun b:boid -> (b.x, b.y) ;;

trans behaviors = {  (* see details on the website *)
  …
  b => (
    let g = barycenter(b) in
    let b' = b + { t = g.t } in
      move_boid(b')
  );



} ;;
```



Multi-Agent Systems

- Reynolds' Boids