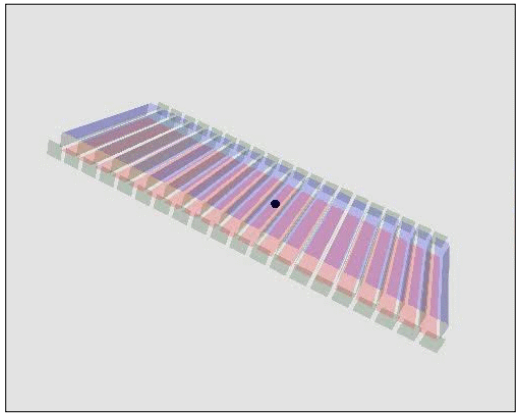
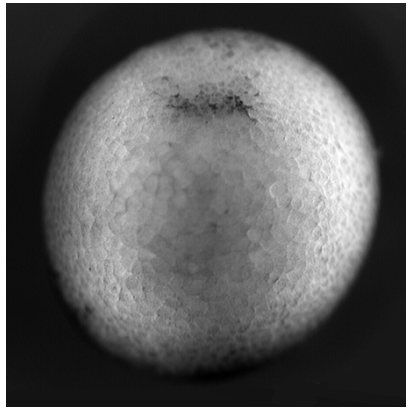
# **MGS** (encore un Modèle Générale pour les Système) : un langage déclaratif pour la programmation spatiale

**Jean-Louis Giavitto**[a]
**Antoine Spicher**[b]
**Olivier Michel**[b]

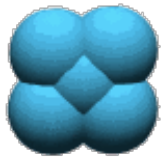
[a] **IRCAM – CNRS**

[b] **LACL – Université de Paris Est**





http://mgs.spatial-computing.org

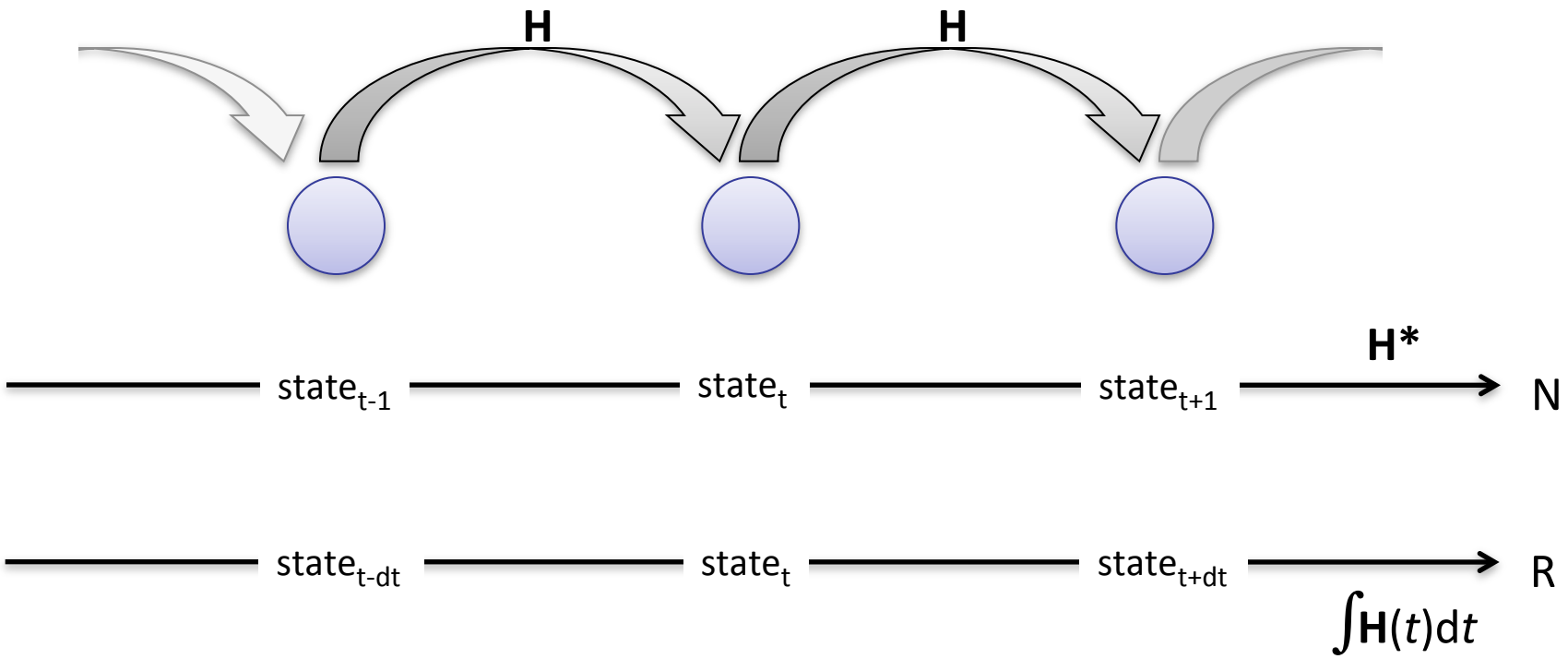


Plusieurs formalismes informatiques tentent de capturer la notion d'interaction dans un système. MGS, un *langage de programmation* expérimental, est fondé sur la constatation que l'ensemble des interactions possibles s'organise suivant une *structure topologique* qui permet de spécifier la description du système et son évolution. Le style de programmation qui en résulte, la *programmation spatiale*, s'appuie sur des relations topologiques (connexité, bord, obstruction...) pour renouveler la notion de structure de données et a trouvé des *applications effectives* dans la modélisation et la simulation de *systèmes dynamiques*, et en particulier pour des systèmes dont la structure varie au cours du temps.

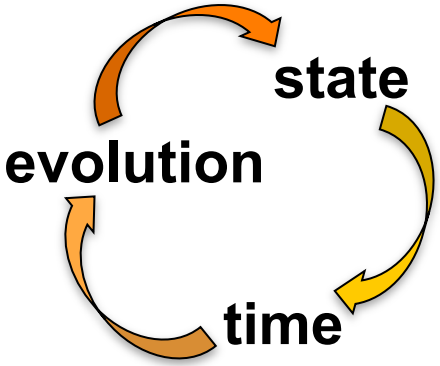
1. $(DS)^2$
2. Une approche spatiale fondée sur les interactions
3. MGS
4. Exemples algorithmiques
5. Modélisations de systèmes biologiques

# Dynamical systems
# and
# Dynamical Structures

# Specifying a dynamical system (for simulation)




**Specification of**
- **structure of state**
- **structure of time**
- **evolution function**



# Formalism for Dynamical System

- State : often structured by space (e.g. fields)
- Time
- Evolution function

| C : continuous, D: discrete | PDE | Coupled ODE | Iteration of functions | Cellular automata | … |
|---|---|---|---|---|---|
| *state* | C | C | C | D | ... |
| *time* | C | C | D | D | ... |
| *space* | C | D | D | D | ... |

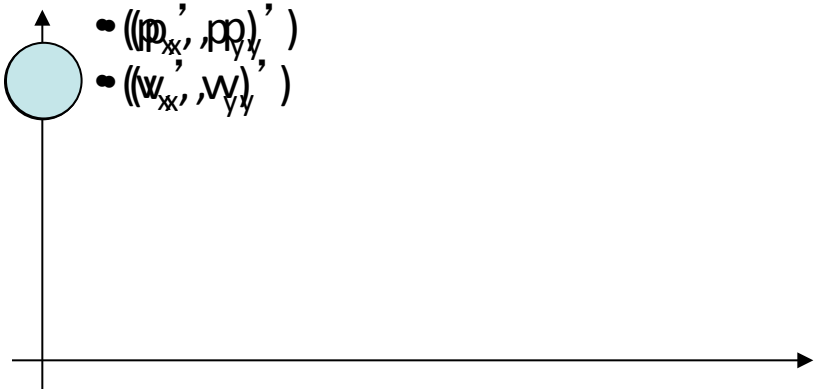
# The medium/process problem

THE CHEMICAL BASIS OF MORPHOGENESIS

By A. M. TURING, F.R.S. *University of Manchester*

(*Received* 9 *November* 1951—*Revised* 15 *March* 1952)

**a falling ball**



*at any time a state is a position and a speed*

A dynamical system (DS)

# The medium/process problem

THE CHEMICAL BASIS OF MORPHOGENESIS

By A. M. TURING, F.R.S. *University of Manchester*

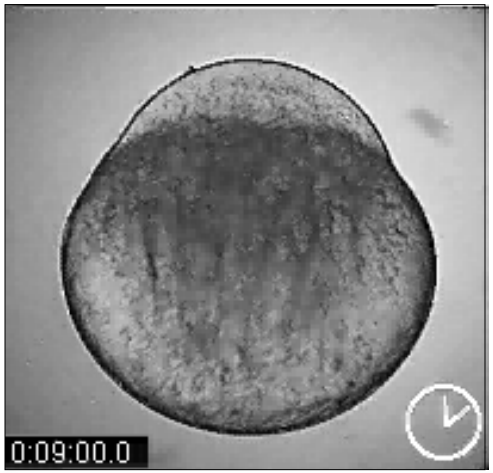
(*Received* 9 *November* 1951—*Revised* 15 *March* 1952)

a falling ball


*at any time a state is a position and a speed*

A dynamical system (DS)

**a developing embryo**




*the structure of the state is changing in time (chemical and mechanical state of each cell)*

**A dynamical system with a dynamical structure $(DS)^2$**

# Modelling morphogenesis: the predefined medium

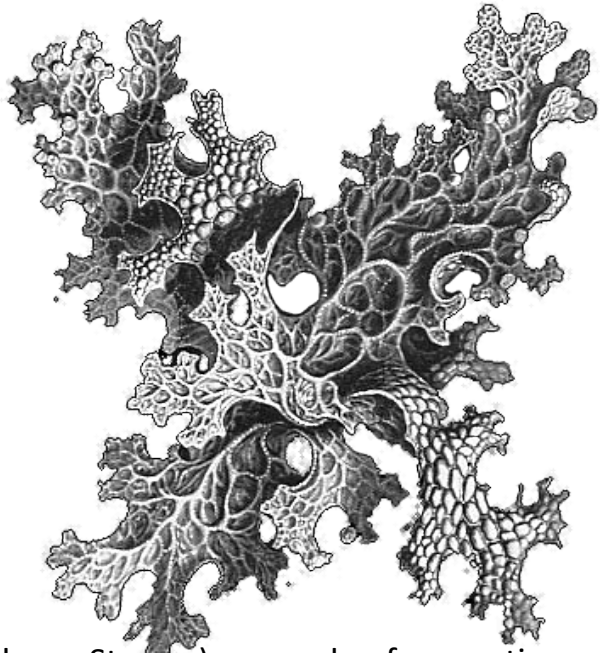
The interdependence of the chemical and mechanical data adds enormously to the difficulty, and attention will therefore be confined, so far as is possible, to cases where these can be separated.

Suppose, for instance, that a 'leg-evocator' morphogen were being produced in a certain region of an embryo, or perhaps diffusing into it, and that an attempt was being made to explain the mechanism by which the leg was formed in the presence of the evocator. It would then be reasonable to take the distribution of the evocator in space and time as given in advance and to consider the chemical reactions set in train by it.

Compatible with
- the notion of morphogenetic field
- cell fate

But
- there is evidence for

**feedback loops between the shape and the process inhabiting the shape**



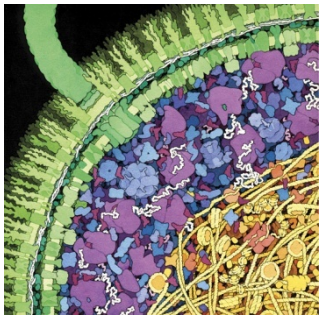
from E. Haenkel (cited by C. Goodman-Strauss): example of a negative curvature surface. Curvature can be controlled while the surface is growing along a 'front'

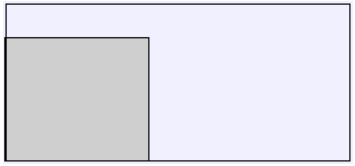
- Dynamical System with a Dynamical Structure
  - Complex systems
  - Whose structure evolves over the time
    - The structure constrains the behavior of the system that modifies the structure …
    - The phase space cannot be defined
    - The evolution function cannot be specified

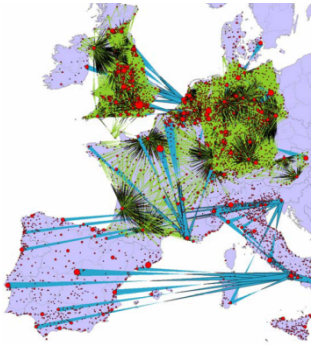
- Examples
  - Biology
    - Molecular biology
    - Developmental biology
  - Physics
    - Soft matter mechanics, multi-scale systems
    - General relativity
  - Urbanism

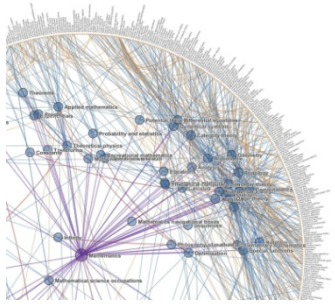
    City growth, traffic control, …
  - Computer science

    Internet, sensor network, reconfigurable robots, software during its entire life cycle, …


© David S. Goodsell 1999


© Tecplot


© L. Sanders - EUROSIM


© C. Harrison - Clusterball project

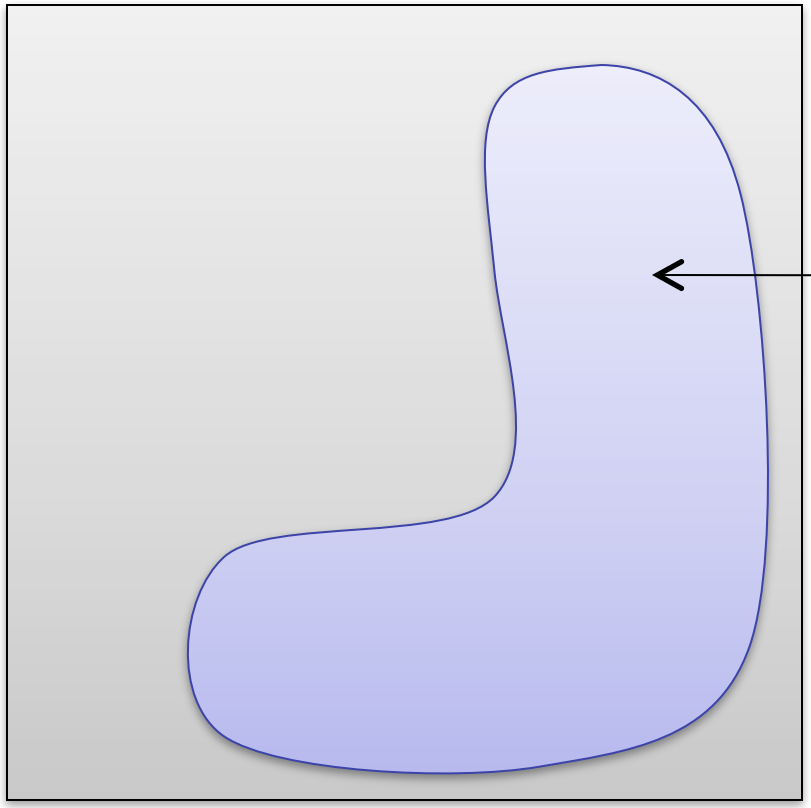
# Comment faire pour modéliser et simuler un $(DS)^2$ ?

- On ne peut pas définir une fonction d'évolution globale
- Le repérage des éléments d'un système est complexe et compliquée
- L'espace des états est difficilement connu

# The Topological Structure of Interactions

# Decompose a system into subsystems following the elements in interaction

A system in some state

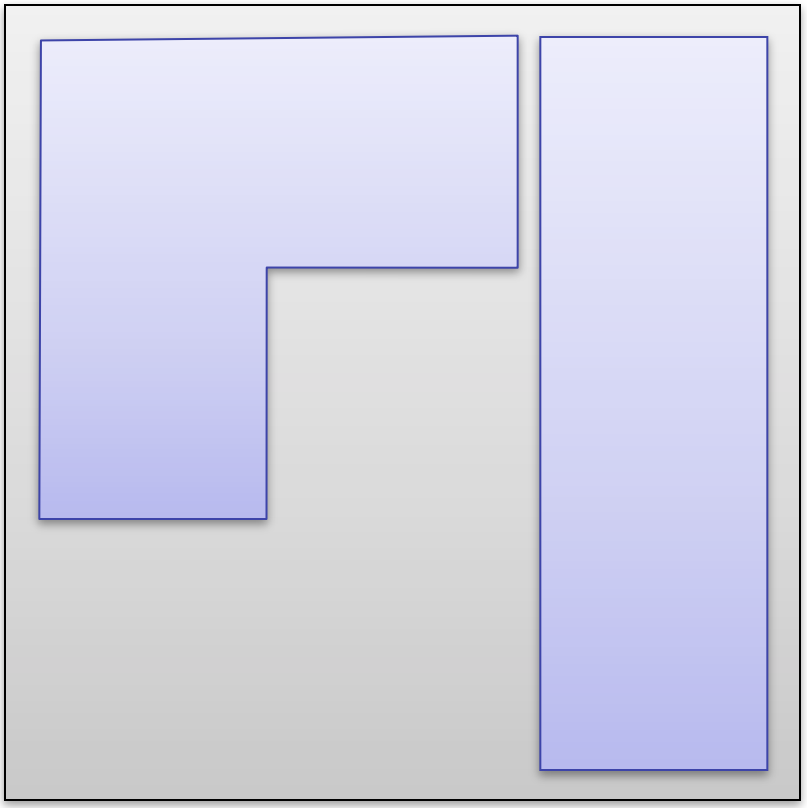


*Part of a system that evolves.*

*Can be identified by comparaison with the previous global state*

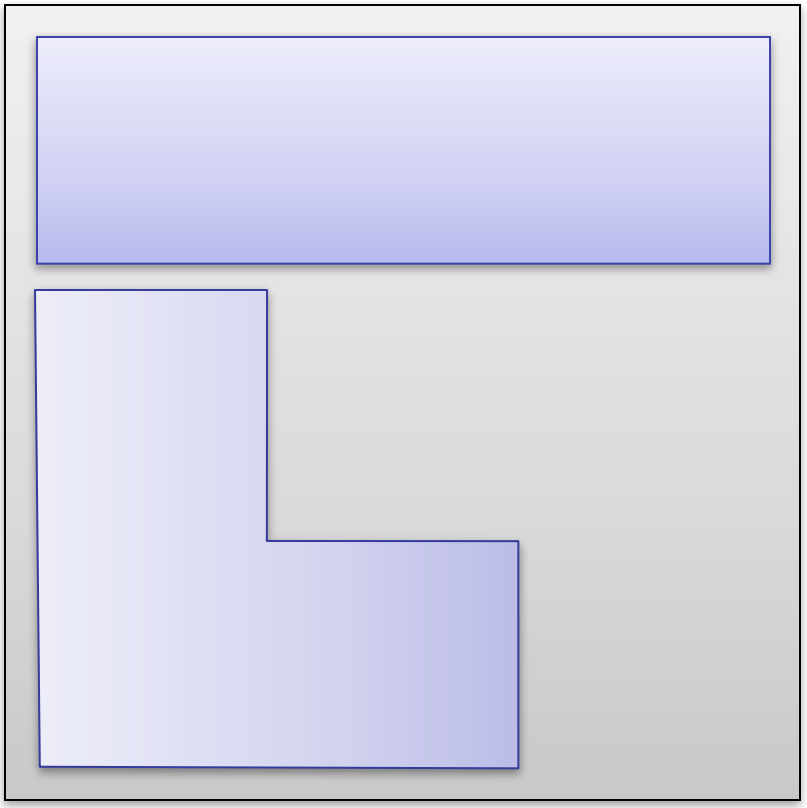
# Decompose a system into subsystems following the elements in interaction

$t = 1$



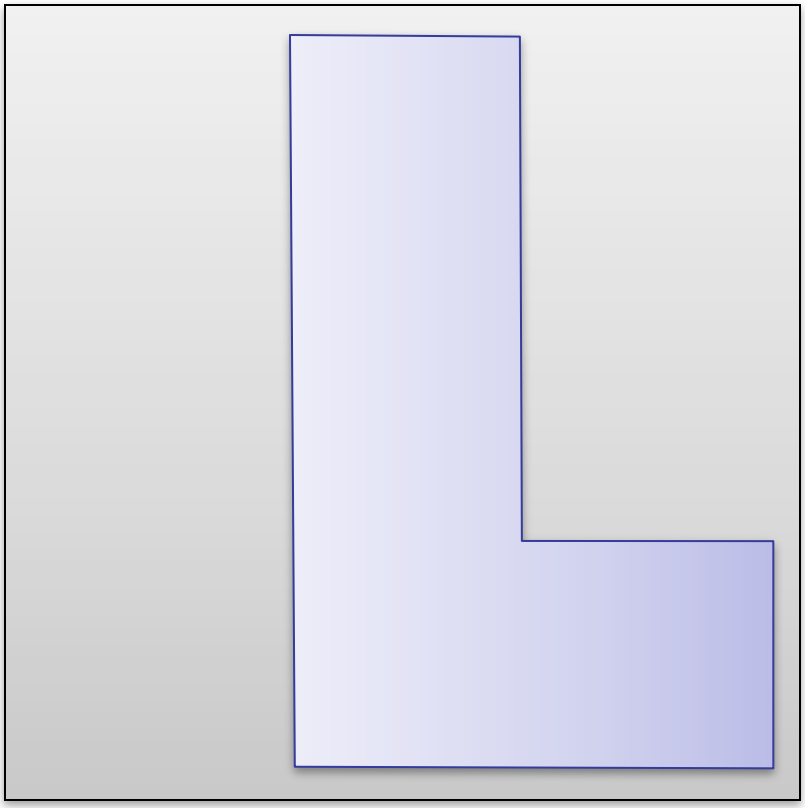
# Decompose a system into subsystems following the elements in interaction

$t = 2$

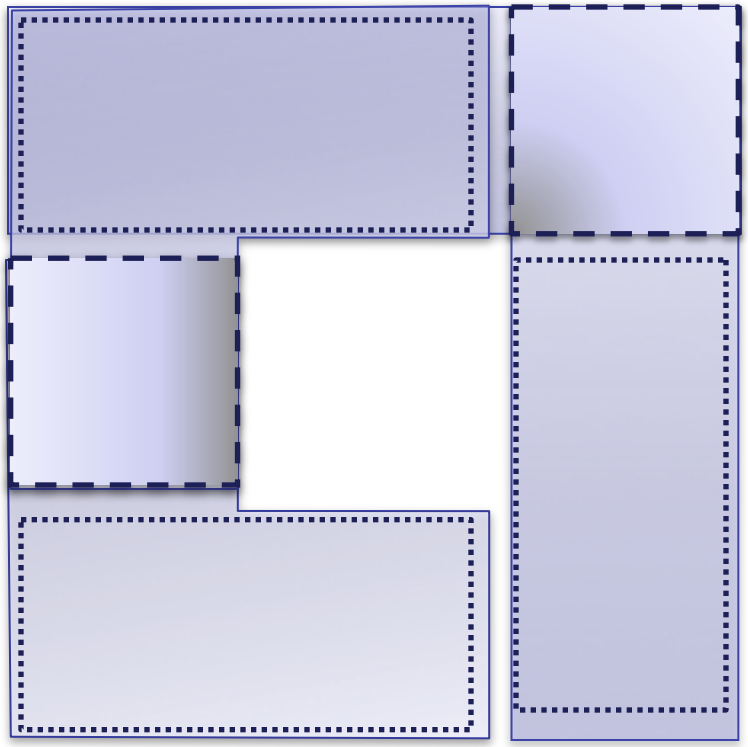


# Decompose a system into subsystems following the elements in interaction

$t = 3$

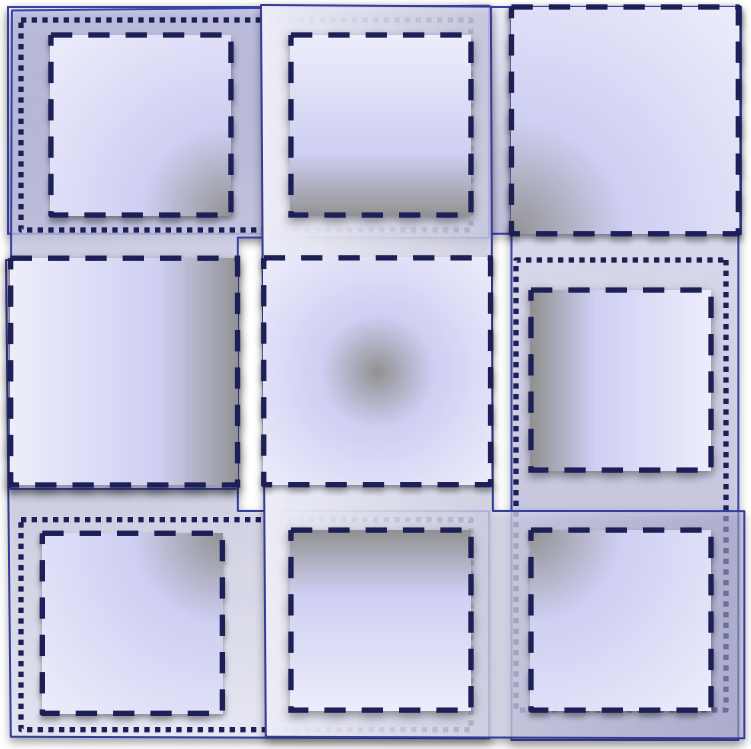


# Decompose a system into subsystems following the elements in interaction

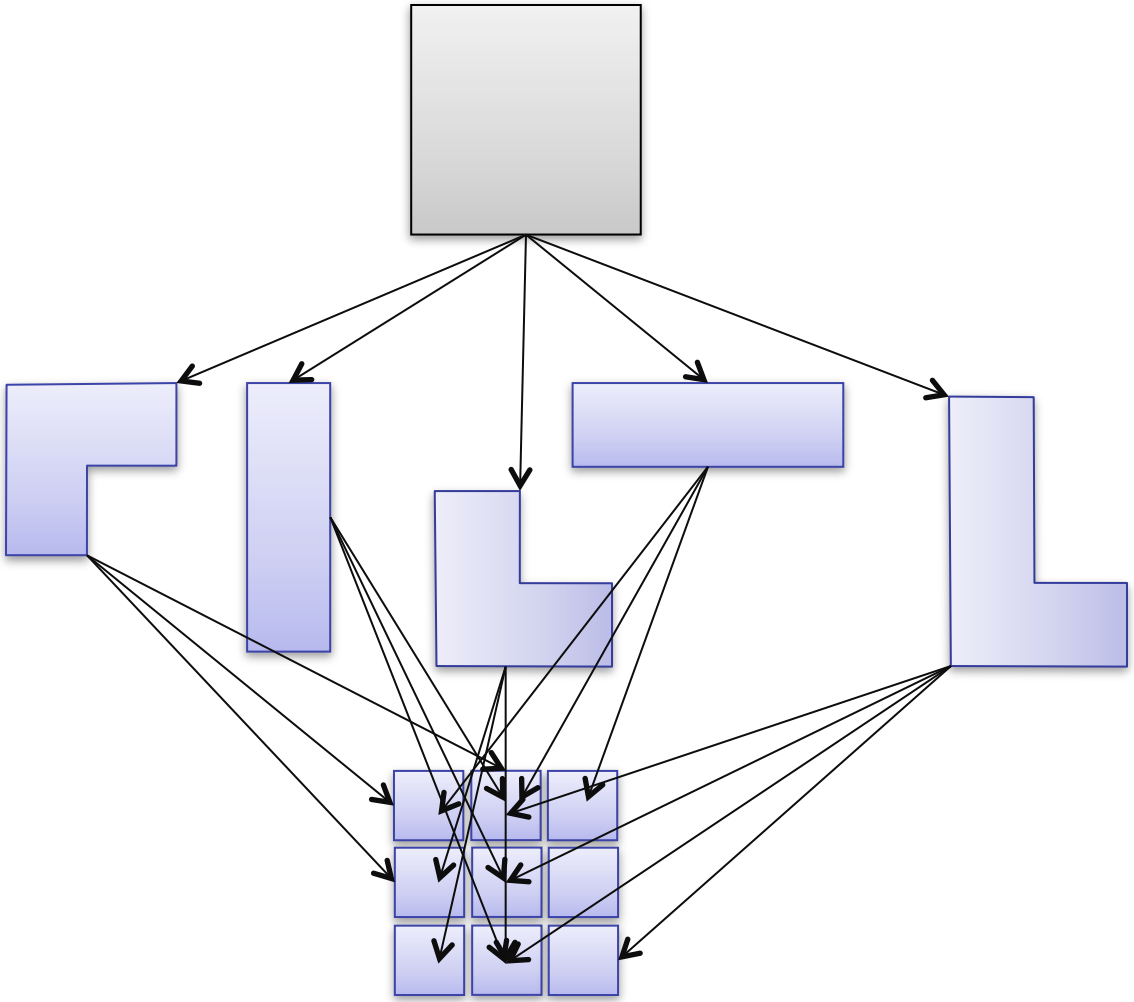


# Decompose a system into subsystems following the elements in interaction

The interactions decomposes the systems into elementary parts.
An interaction implies one or several elementary parts.



# Decompose a system into subsystems following the elements in interaction



the inclusion structure between the elementary and interacting parts is **a lattice**

**a (simplicial) complex** is a better (topological) equivalent representation

MGS

## The big picture

1. *Develop concepts and tools to define "topological collections"*
   - *Space build using subspaces*
   - *Labeled by values (~ fields in physics)*

2. *Develop concepts and tools to specify interacting parts and computes in these spaces*

3. *Develop Applications*

# Topological rewriting

&

# MGS

# The MGS project

- Language dedicated to the simulation of $(DS)^2$
- Declarative (declarative simulation *vs* procedural)
- Abstract rewriting of complex spatial structures:
  - Data structure = topological collections

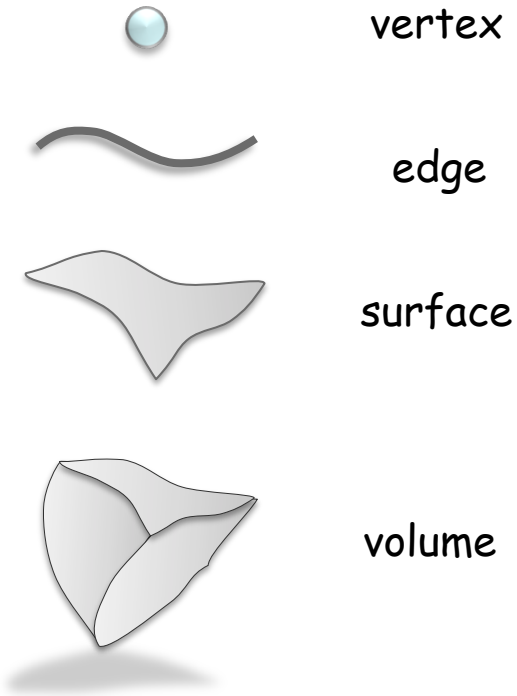
    sequence, generalized array, (multi-)set, arbitrary graph, Delaunay triangulation, g-map, …, cell complexes
  - Control structure = transformation
    - two powerful languages to specify sub-collections (elements in interaction)
    - Various rule application strategies: maximal parallel, asynchronuous, stochastic, Gillespie-like, …

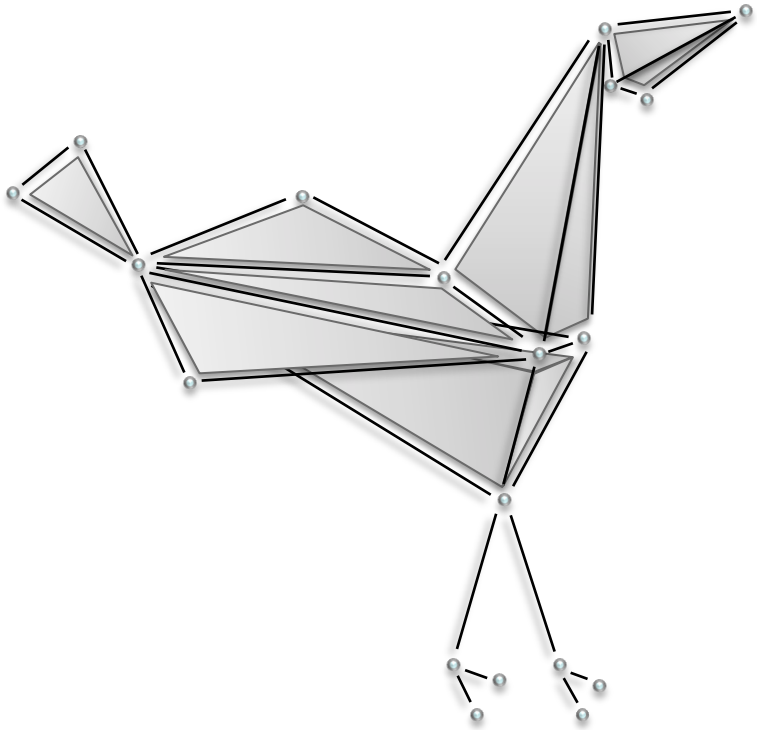
# Properties



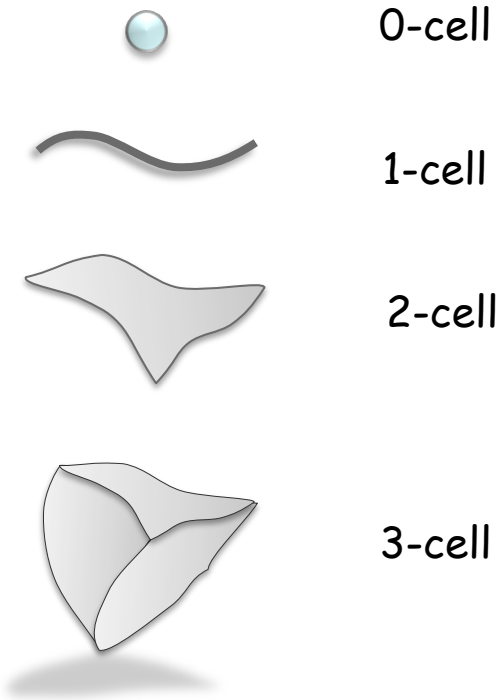
- *local evolution rules*
  mandatory when you cannot express a global function/relation because the domain of the function/relation is changing in time
- interaction based approach
  the l.h.s. of a rule specifies a set of elements in *interaction*, the r.h.s. the result of the interaction
- *the phase space is well defined but not well known*
  a generative process enumerates the elements but membership-test can be very hard
- *various kind of time evolution* (for the same set of rules)
- *demonstration by induction*
  on the rules or on the derivation (e.g. growth function in L system)

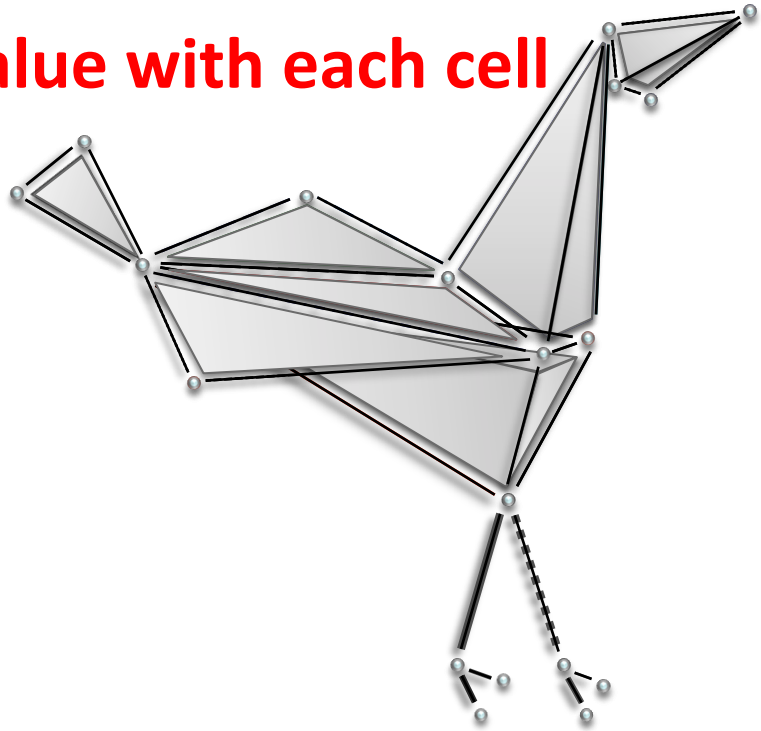
- Topological collections
  - Structure
    - A collection of topological cells
    - An ***incidence relationship***



- Topological collections
  - Structure
    - A collection of topological cells
    - An incidence relationship
  - Data: **association of a value with each cell**

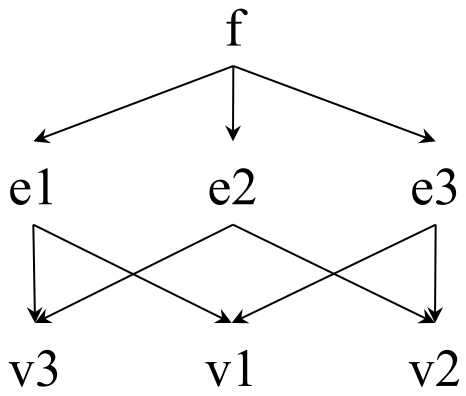


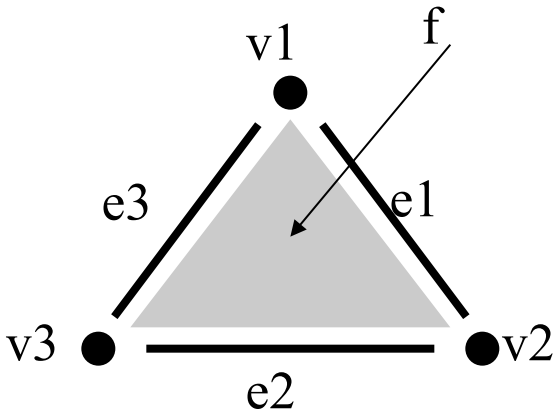



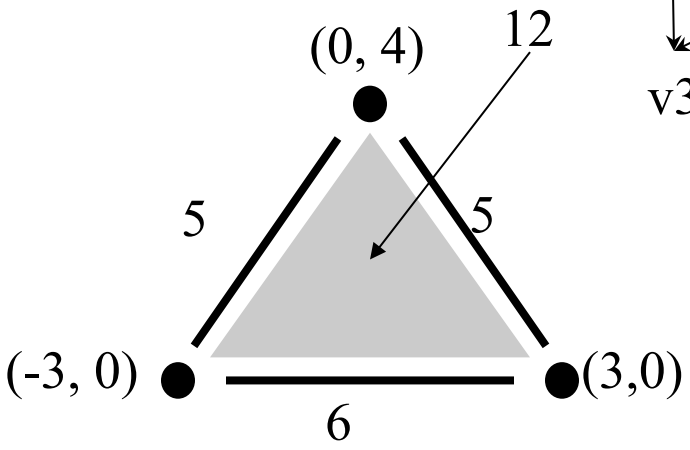
# Abstract (Simplicial) Complex and (Simplicial) Chains

*Incidence relationship and lattice of incidence:*

- boundary(f) = {v1, v2, v3, e1, e2, e3}
- faces(f) = {e1, e2, e3}
- cofaces(v1) = {e1, e3}



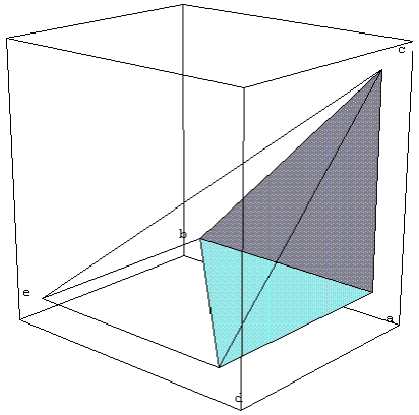




*Topological chain*

- coordinates with vertices
- lengths with edges
- area with f

$$\begin{pmatrix}0\\4\end{pmatrix}.v_1+\begin{pmatrix}3\\0\end{pmatrix}.v_2+\begin{pmatrix}-3\\0\end{pmatrix}.v_3+5.e_1+6.e_2+5.e_3+12.f$$

# Topology ↔ simplicial complex↔ lattice

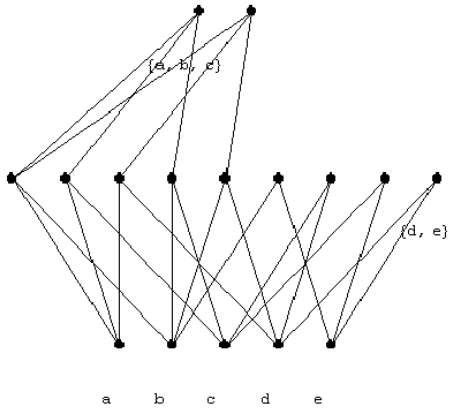



## Topology (open and closed sets)

↓ simplices = Closed sets ↑

## Simplicial Complex (set of sets closed by inclusion/intersection)

↓ simplices = Lattice cones ↑




## Lattice (order relation: ∧ , ∨)

- Concise reformulation of classical approaches
- Extension

- Transformations
  - Functions defined by case on collections
    Each case (pattern) matches a sub-collection
  - Defining a rewriting relationship: ***topological rewriting***

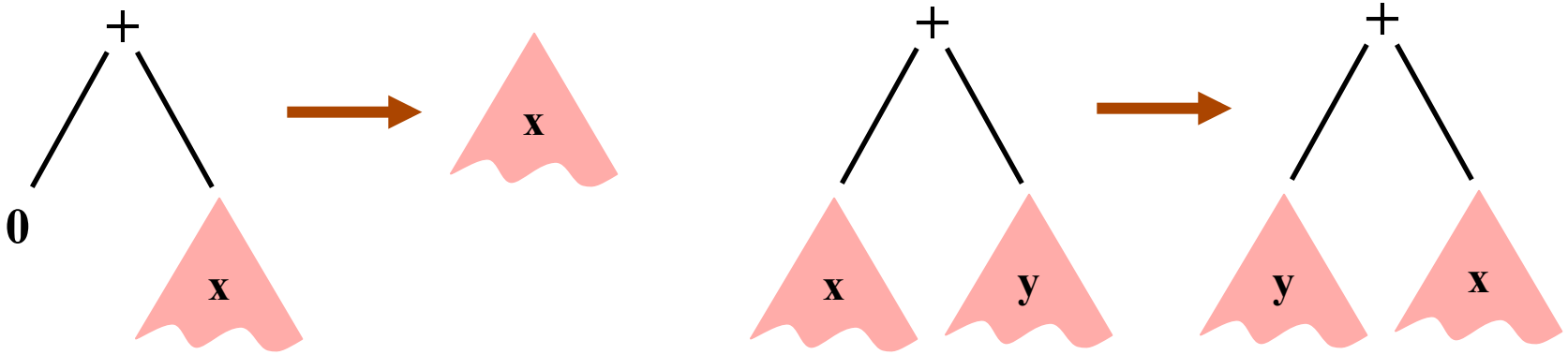
**trans** *T* = **{**
    $pattern_1 \Rightarrow expression_1$
    …
    $pattern_n \Rightarrow expression_n$
**}**

# **Rewriting systems** (and abstract transition systems)

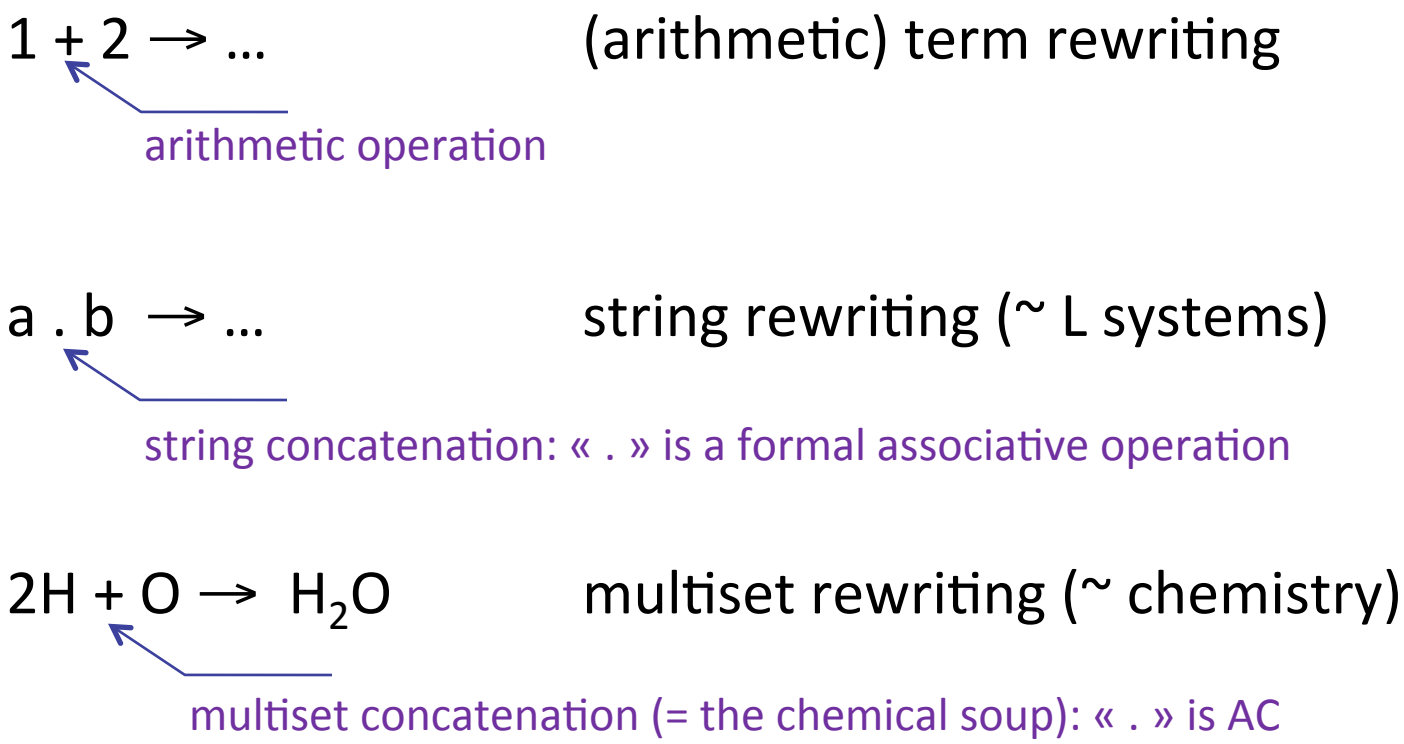
MGS

- Rewriting system
  - Used to formalize equationnal reasoning
  - A generative device (grammar)
  - Replace a sub-part of an entity by an other
  - Set of rewriting rules $\alpha \rightarrow \beta$
    - $\alpha$: pattern specifying a sub-part
    - $\beta$: expression evaluating a new sub-part

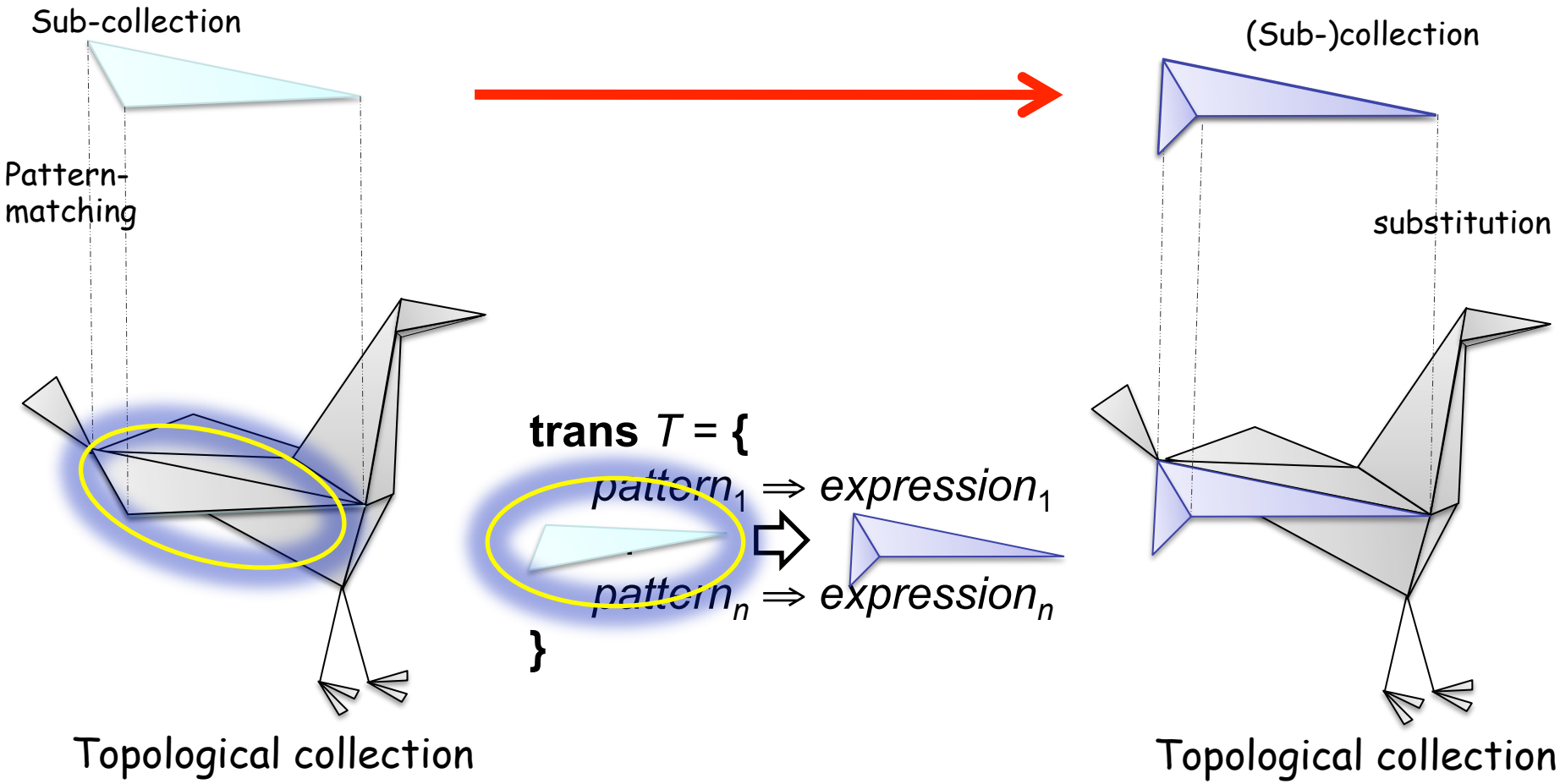
- Example: arithmetic expressions simplification



# Rewriting on terms+property



- Transformations



# Topological rewriting = transformation

$1 + 2 \rightarrow \ldots$ (arithmetic) term rewriting

arithmetic operation

$a \,.\, b \rightarrow \ldots$ string rewriting (~ L systems)

string concatenation: « . » is a formal associative operation

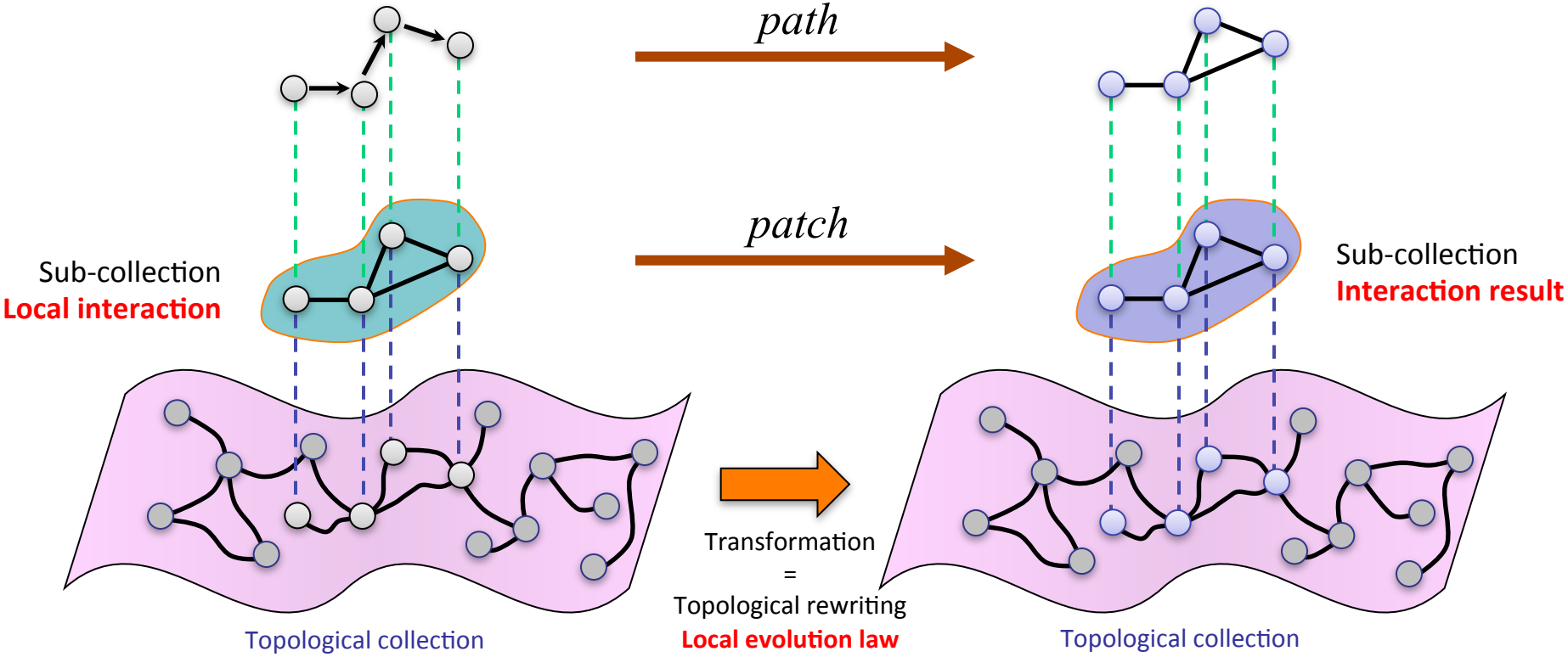
$2H + O \rightarrow H_2O$ multiset rewriting (~ chemistry)

multiset concatenation (= the chemical soup): « . » is AC

$v_1.\sigma_1 + v_2.\sigma_2 \rightarrow \ldots$ topological rewriting (MGS)

gluing cell in a cell complex: … (AC and algebraic machinery)

# Transformation




**Pattern matching : specifying a sub-collection of elements in interaction**

- *Path transformation* (path = sequence of neighbor elements)
  - Concise but limited expressiveness
- *Patch transformation* (arbitrary shape)
  - Longer but higher expressiveness

# Example: Diffusion Limited Aggregation (DLA)

- Diffusion: some particles are randomly diffusing; others are fixed
- Aggregation: if a mobile particle meets a fixed one, it stays fixed

```
trans dla = {
    `mobile , `fixed   => `fixed, `fixed ;
    `mobile , <undef> => <undef>, `mobile
}
```

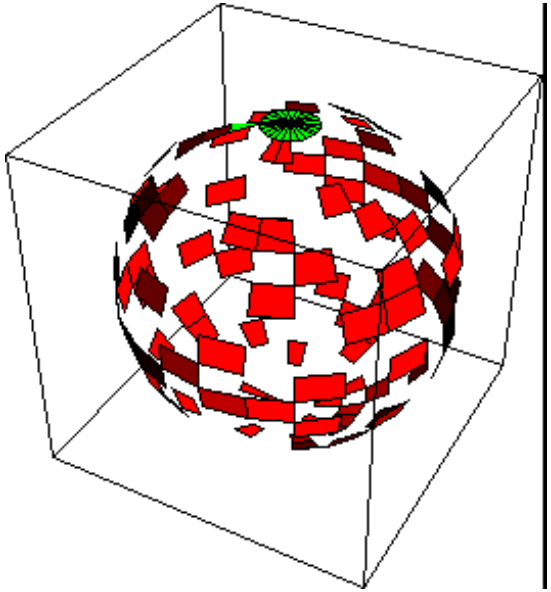
NEIGHBOR OF

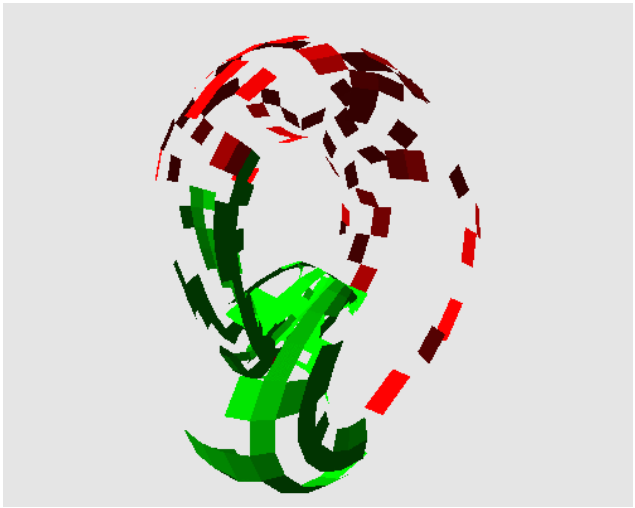
# Example: Diffusion Limited Aggregation (DLA)

- Diffusion: some particles are randomly diffusing; others are fixed
- Aggregation: if a mobile particle meets a fixed one, it stays fixed

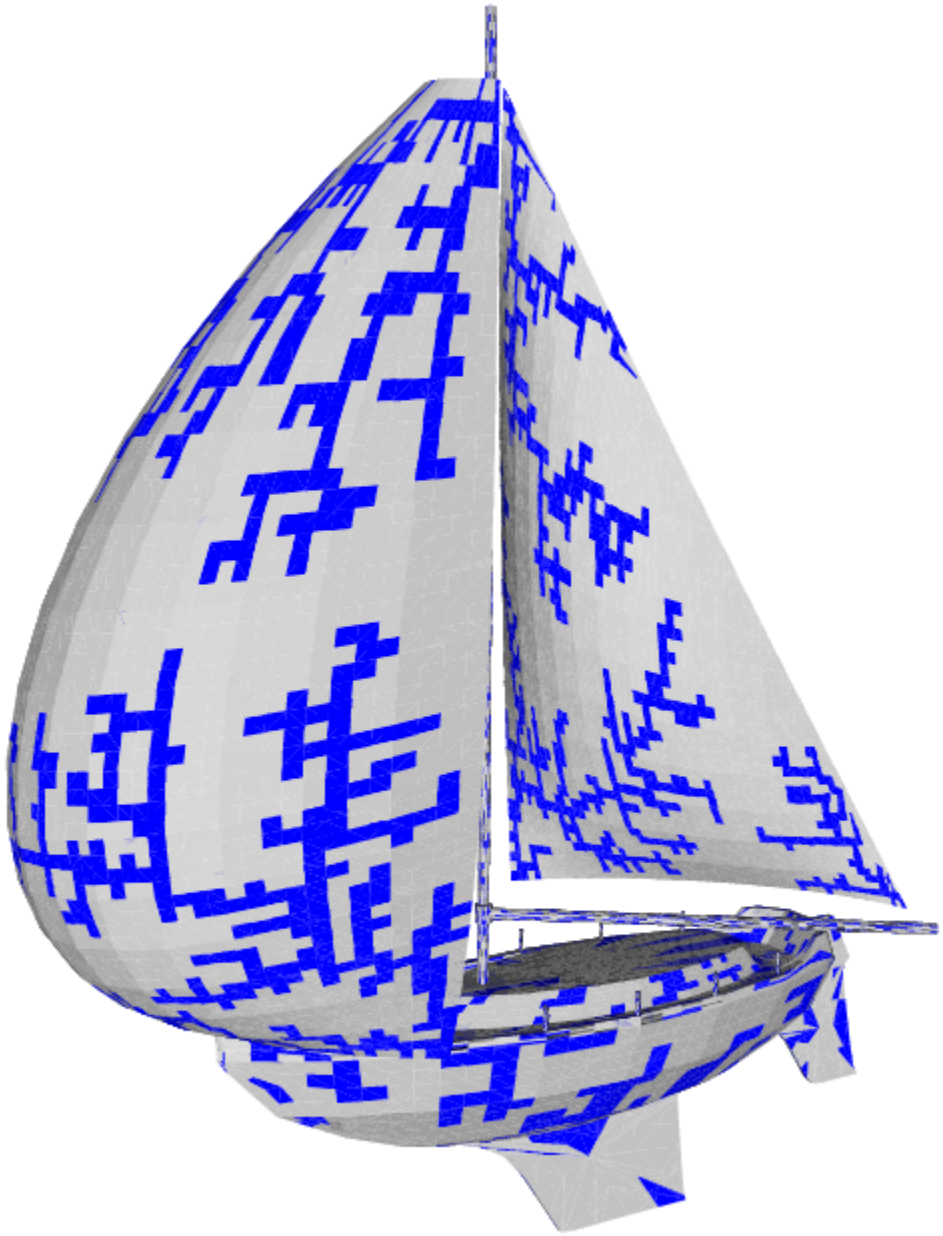
```
trans dla = {
    `mobile , `fixed  => `fixed, `fixed ;
    `mobile , <undef> => <undef>, `mobile
}
```

this transformation is an abstract process that can be applied to any kind of space





# Polytypisme



# The Spatial Approach



- Use space (topology) to unify the various collection structures
  - space as as a resource
  - space as a constraint
  - space as an input/output
- Neighborhood relationships:
  - the structure of the collection
  - the structure of the subcollection
  - the computation dependencies
- Substitution (replacement) topological surgery

MGS

# Complex systems←→Rewriting techniques

## Modelling

**State (space)**

hierarchical and tree organizations

*arbitrary complex organizations*

**Evolution function**

***interactions*** → evolution

***local* evolution laws**

## Specification

**Data structure**

formal trees (or terms)

**?**

**Set of rules**

α : pattern → β : expression

rewriting rules

## Simulation

Trajectories

Time management

discrete, event-based,
synchronous vs. asynchronous

## Application

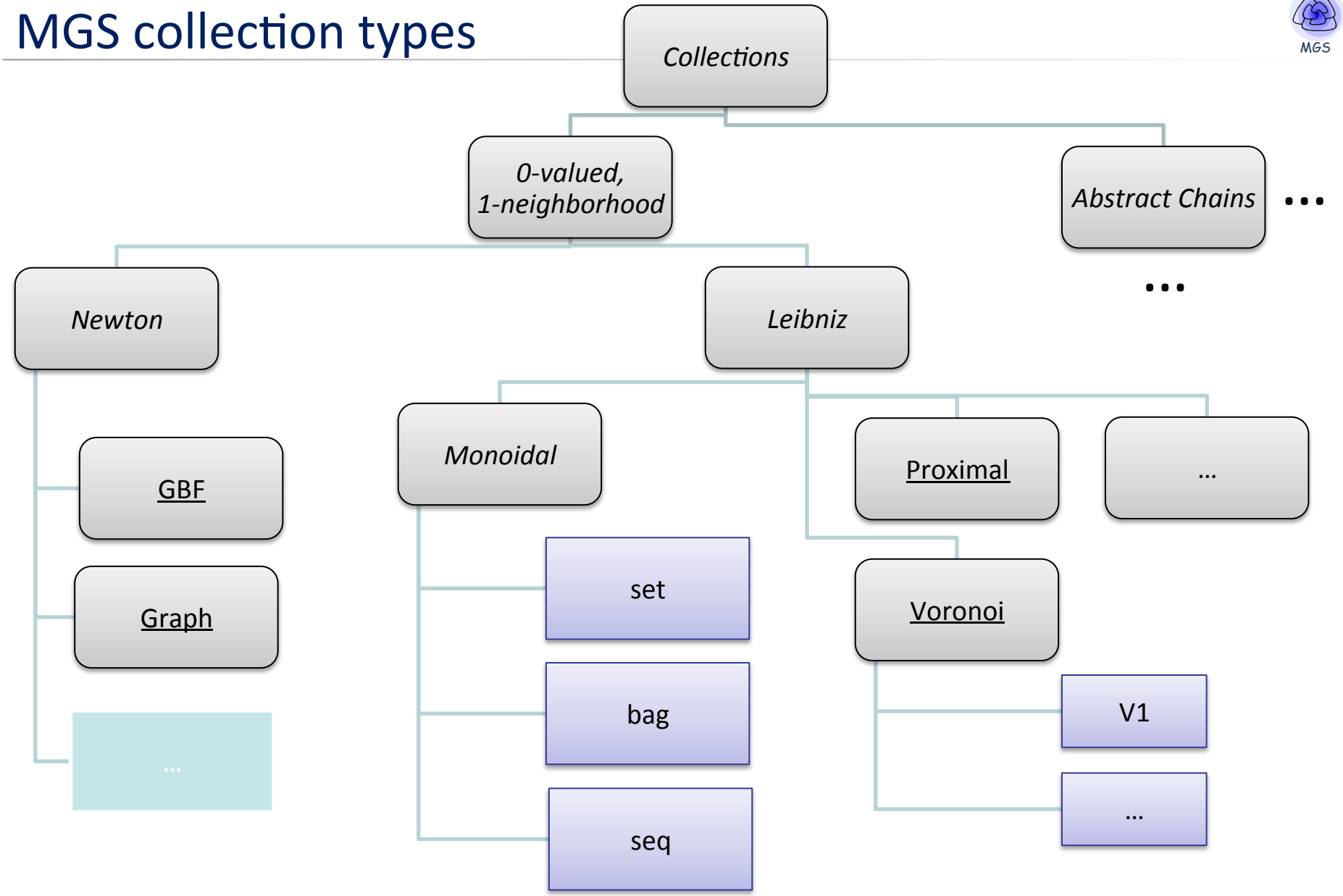
Derivations

Rule application strategy

maximal parallel, sequential,
deterministic, stochastic

# Algorithmic examples

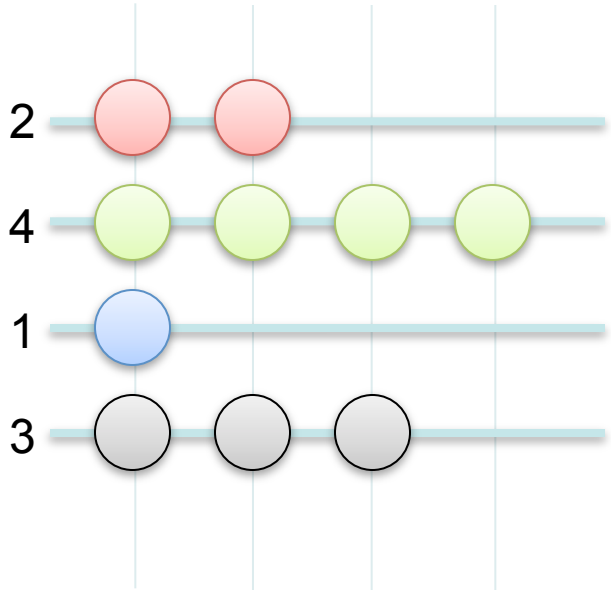
# MGS collection types



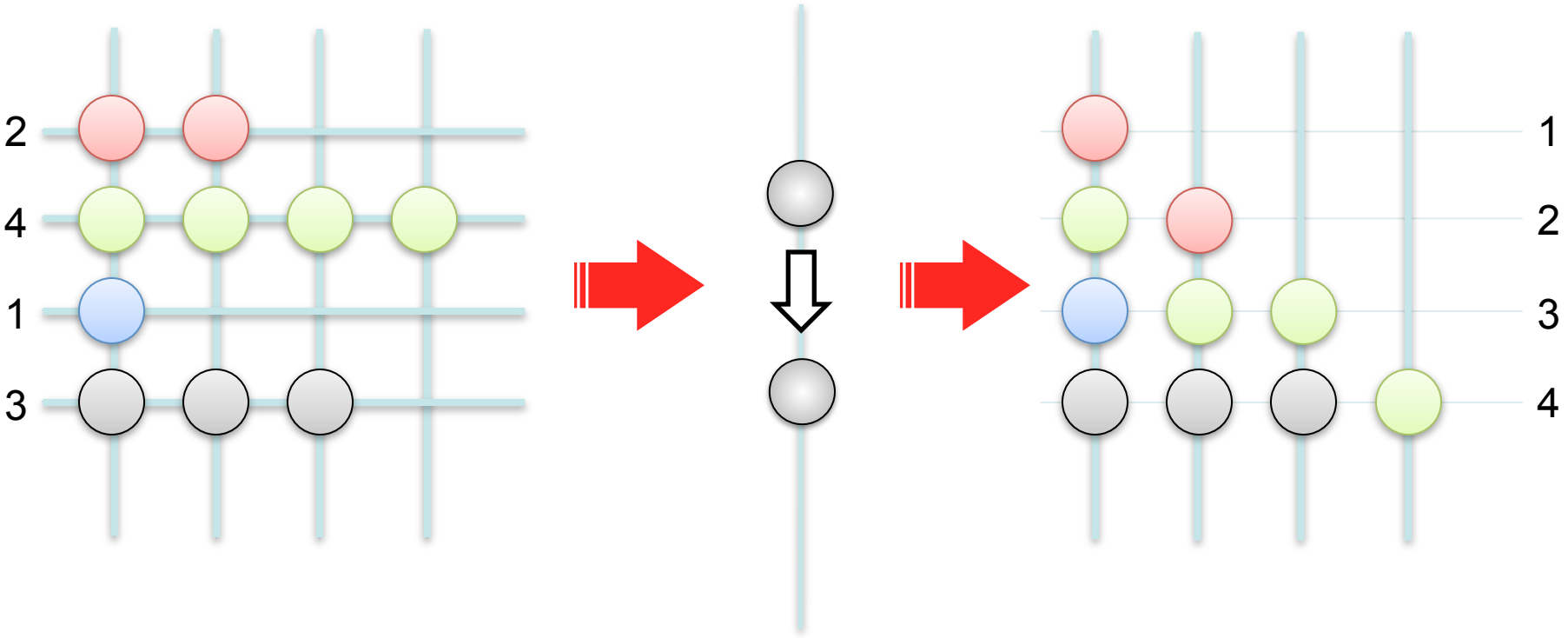

Leibniz: `x => <undef>` means delete x
Newton: `x => <undef>` means an undefined value @ x

*Abstract type* | Type constructor | Concrete type

# Bead Sort



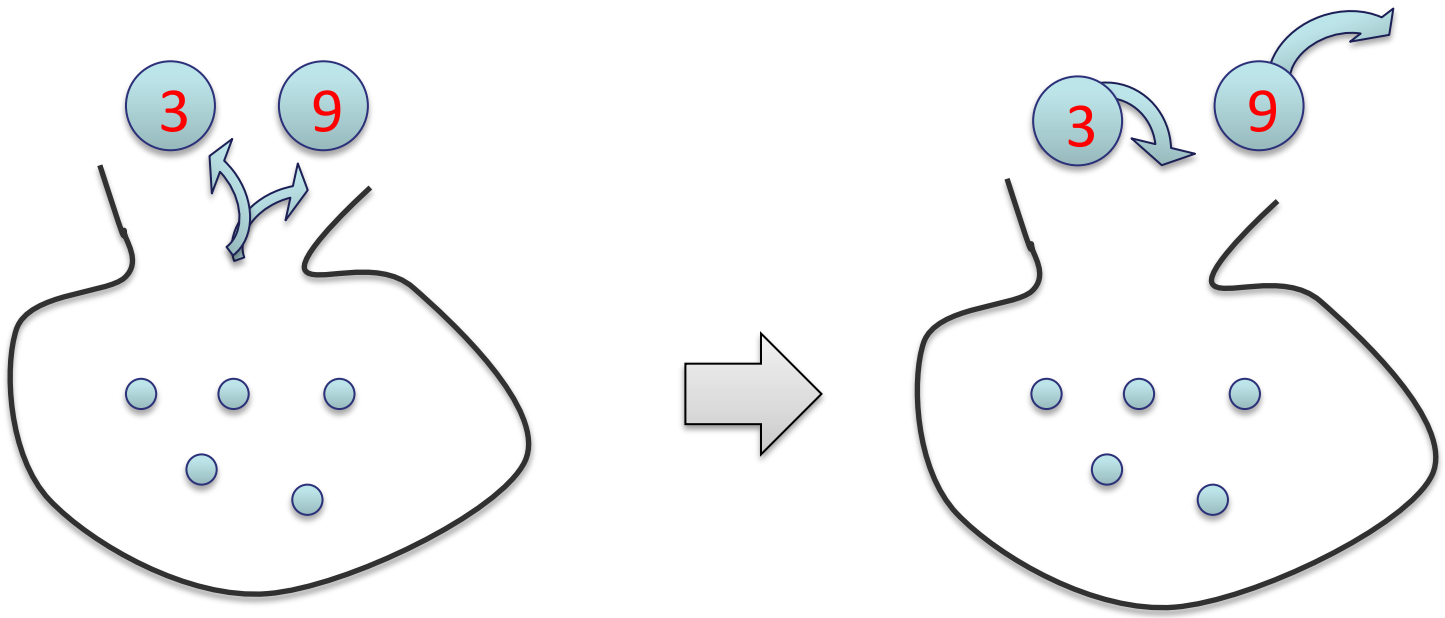
# Bead Sort



# Bead Sort

```
Gbf NEWS = < North, South, East, West;
             North+South=0, East+West=0>

trans dla = {
    `bead |south> `empty  => `empty, `bead ;
}
```

# Eratosthene's Sieve

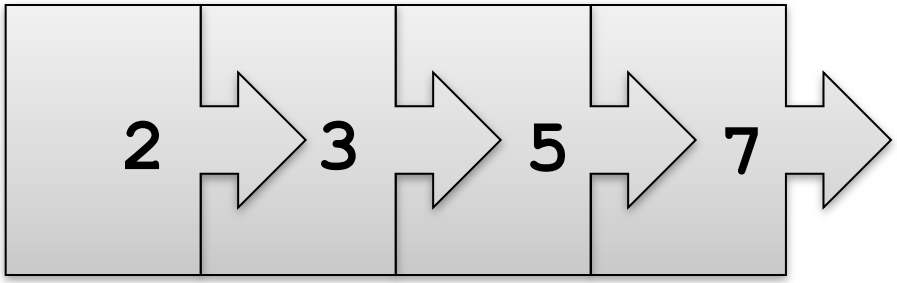



$$\begin{aligned}
\texttt{trans}\ \mathit{Generate} &= \{x, true\} => x, \{x+1, true\}; \\
\texttt{trans}\ \mathit{Succed} &= \{x, true\} => x; \\
\texttt{trans}\ \mathit{Eliminate} &= (x, y \,/\, y \bmod x = 0) => x;
\end{aligned}$$

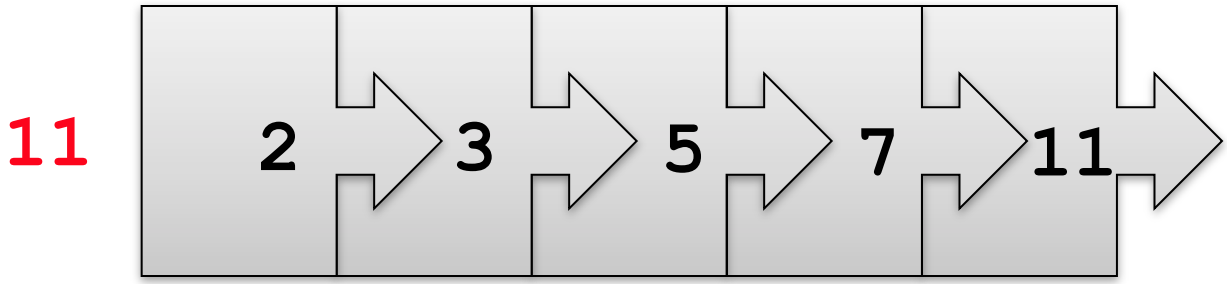
$$\mathit{Eliminate}[\texttt{fixrule}]\Big(\mathit{Succed}\big(\mathit{Generate}[N](\{2, true\}, \texttt{set} : ())\big)\Big)$$

# Eratosthene's Sieve

9

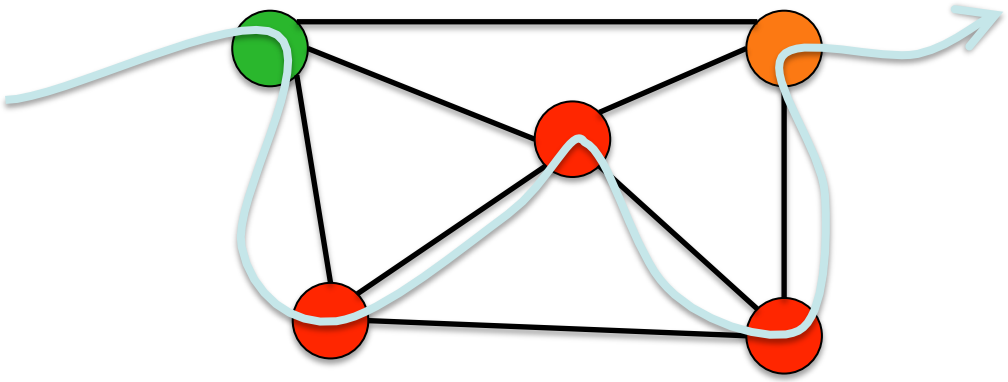


# Eratosthene's Sieve

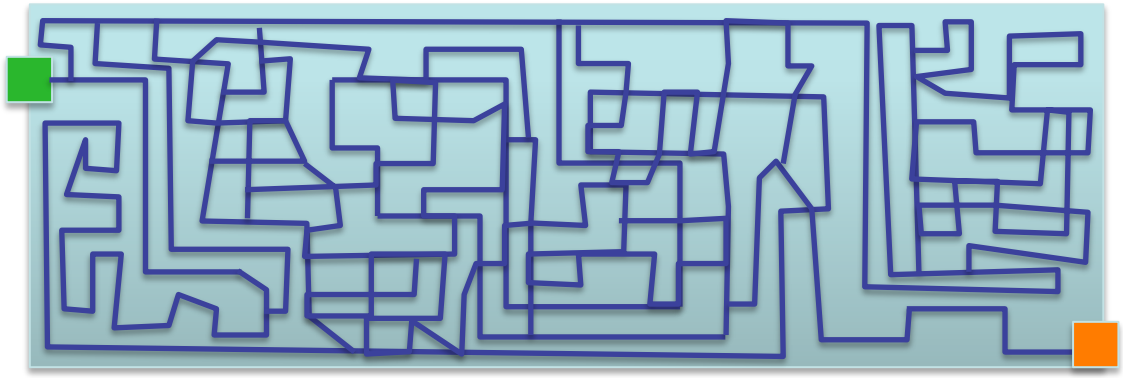



$$
\begin{array}{l}
\texttt{trans}\ Eratos = \{ \\
\qquad Genere1 \;=\; n : integer \;/\; \tilde{}\texttt{right}\ n \;\Rightarrow\; n,\ \{prime = n\}; \\
\qquad Genere2 \;=\; n : integer, \{prime\ \texttt{as}\ x, \tilde{} candidate, \tilde{} ok\} \\
\qquad\qquad \Rightarrow\; n+1,\ \{prime = x, candidate = n\}; \\
\qquad Test1 \;=\; \{prime\ \texttt{as}\ x, candidate\ \texttt{as}\ y, \tilde{} ok\} \;/\; y \bmod x = 0 \;\Rightarrow\; \{prime = x\}; \\
\qquad Test2 \;=\; \{prime\ \texttt{as}\ x, candidate\ \texttt{as}\ y, \tilde{} ok\} \;/\; y \bmod x <> 0 \\
\qquad\qquad \Rightarrow\; \{prime = x, ok = y\}; \\
\qquad Next \;=\; \{prime\ \texttt{as}\ x1, ok\ \texttt{as}\ y\}, \{prime\ \texttt{as}\ x2, \tilde{} ok, \tilde{} candidate\} \\
\qquad\qquad \Rightarrow\; \{prime = x1\}, \{prime = x2, candidate = y\}; \\
\qquad NextCreate \;=\; \{prime\ \texttt{as}\ x, ok\ \texttt{as}\ y\}\ \texttt{as}\ s \;/\; \tilde{}\texttt{right}\ s \\
\qquad\qquad \Rightarrow\; \{prime = x\}, \{prime = y\}; \\
\}
\end{array}
$$

# Hamiltonian path

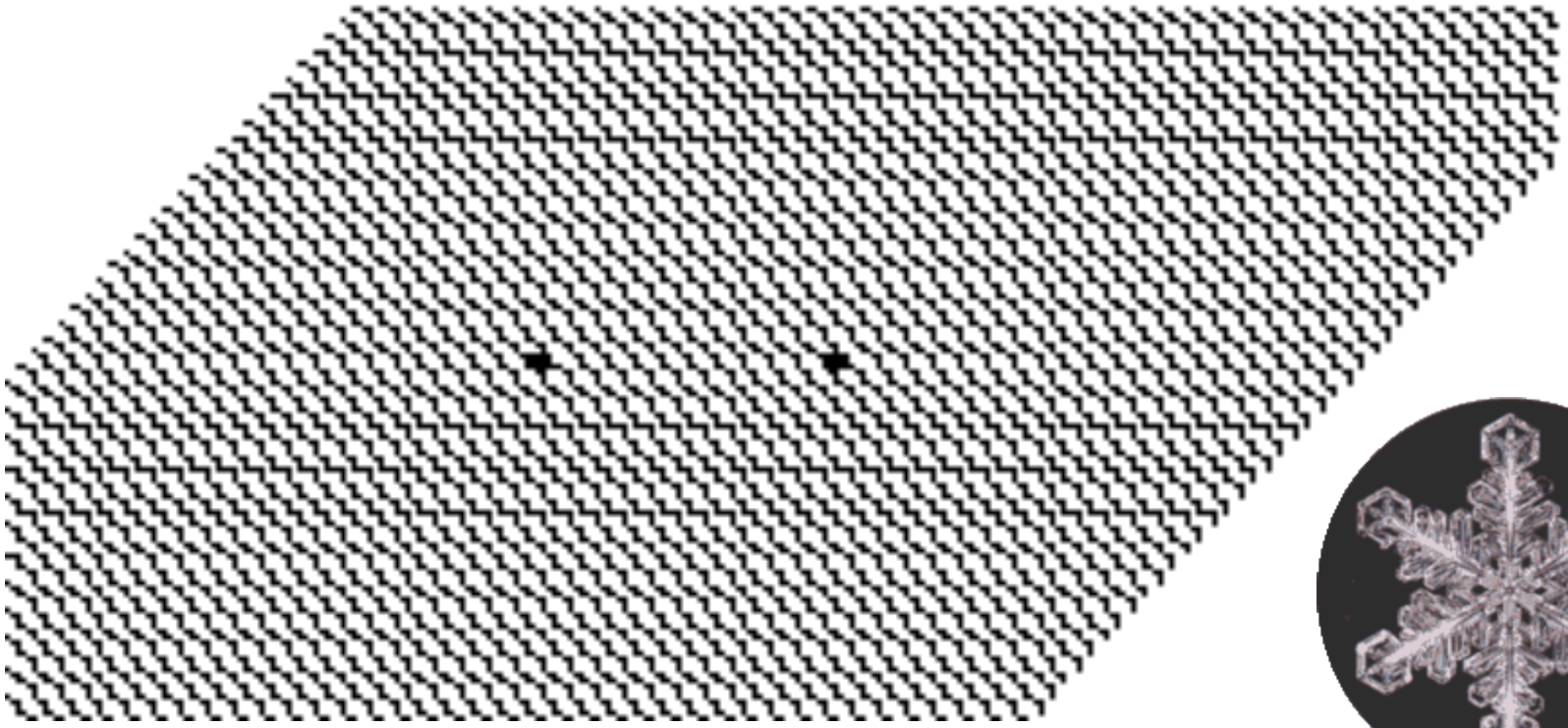


```
trans h_path = { `start , x* as p,`stop
                 / size(p) = n-2 => return p }
```



```
trans maze = { `input, c* as p,`output => return p }
```

# Snowflake formation (CA-based)

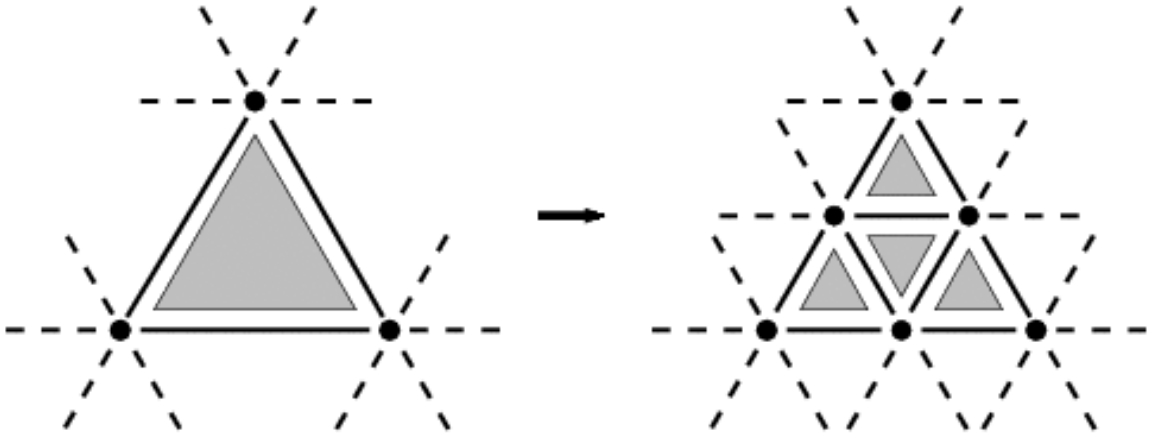




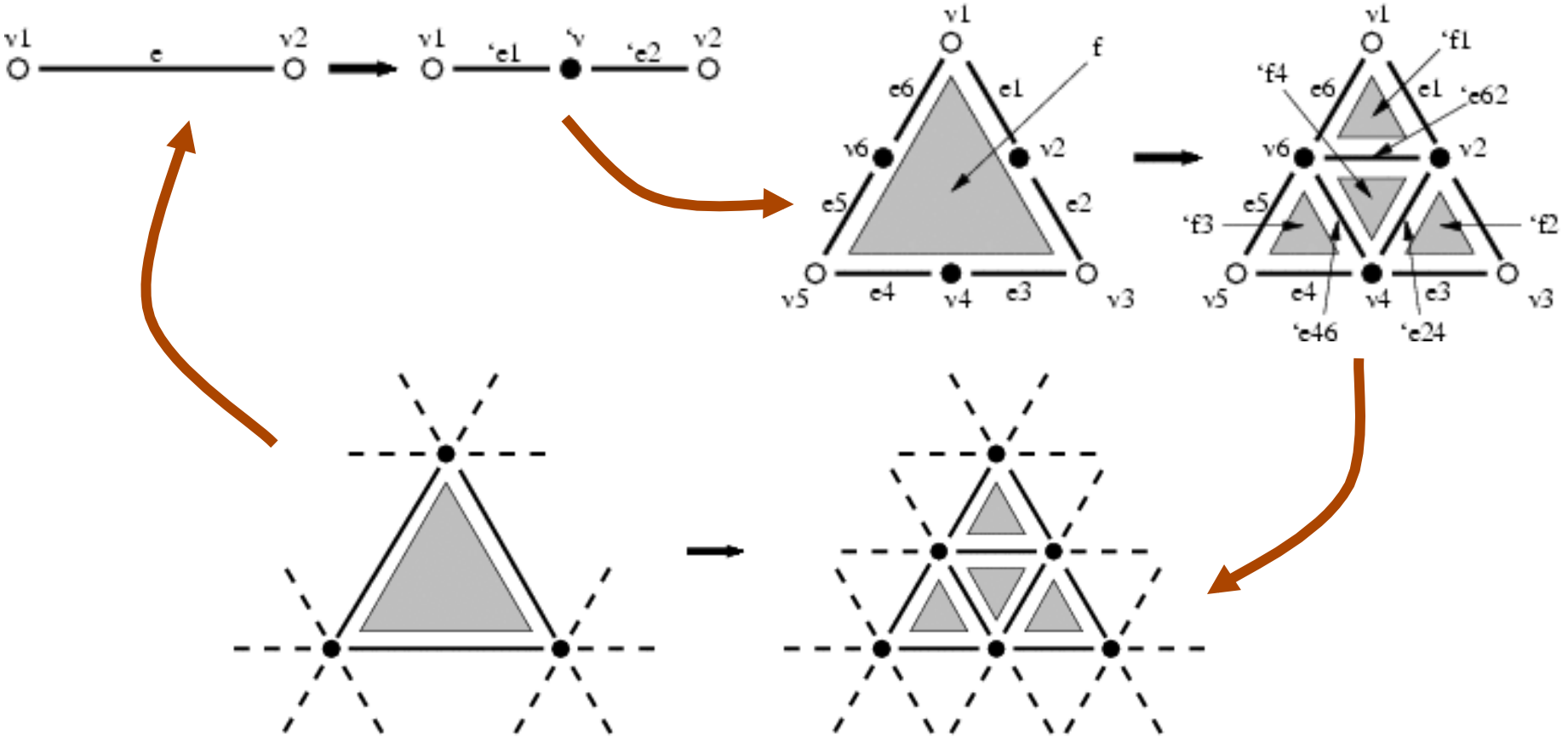
```
gbf hexa = <a, b, c;  a+b=c>

trans T = {
    (0 as x / (neighborsfold(+, 0, x)==1)
     => 1)
}
```

# Surface subdivision

- Surface refinement by mesh subdivision
- Intuitive to describe (locally) but not trivial to implement
- Polyhedral subdivision
  - Modifying the neighborhood
  - 2 steps: edge subdivision and triangles building



# Surface subdivision



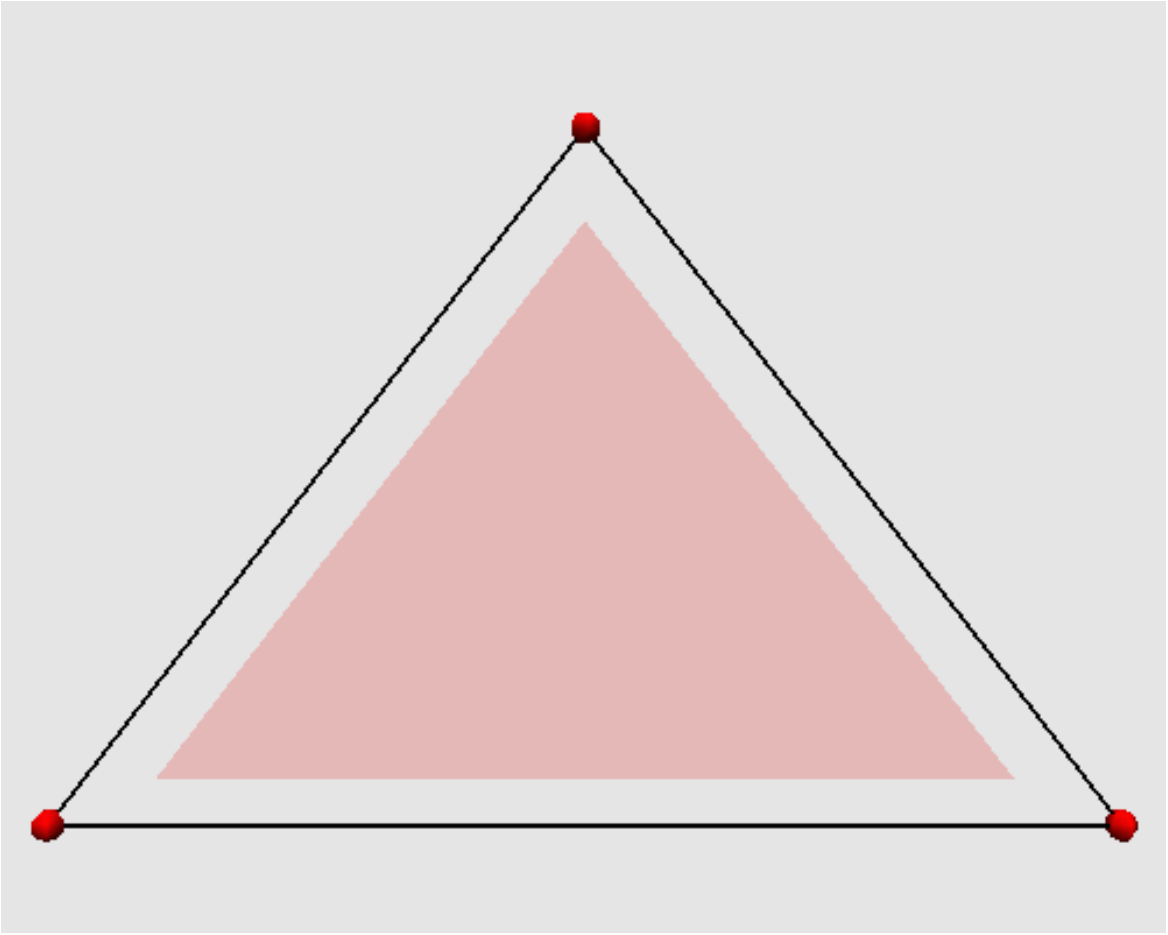
# Patch transformation

- Example: surface subdivision

```
patch insert_vertex[genNb] = {
  ~v1 < e:[dim=1] > ~v2 =>
    `e1:[dim=1,faces=(v1,`v),val=`edge]
    `v: [dim=0,cofaces=(`e1,`e2),val={gen=genNb,x=…}]
    `e2:[dim=1,faces=(v2,`v),val=`edge]
} ;;
```

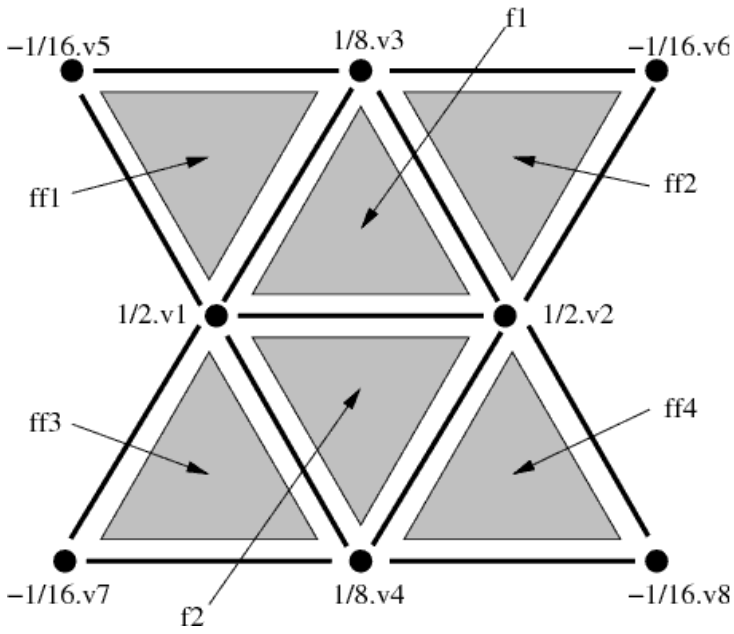
```
patch build_triangles[genNb] = {
  f:[dim=2,faces=(e1,e2,e3,e4,e5,e6)]
  ~v1 < ~e1 > ~v2:[v2.gen==genNb] < ~e2 >
  ~v3 < ~e3 > ~v4:[v4.gen==genNb] < ~e4 >
  ~v5 < ~e5 > ~v6:[v6.gen==genNb] < ~e6 > ~v1 =>
    `e24:[dim=1,faces=(v2,v4),val=`edge]
    `e46:[dim=1,faces=(v4,v6),val=`edge]
    `e62:[dim=1,faces=(v6,v2),val=`edge]
    `f1: [dim=2,faces=(e6,e1,`e62),val=`face]
    `f2: [dim=2,faces=(e2,e3,`e24),val=`face]
    `f3: [dim=2,faces=(e4,e5,`e46),val=`face]
    `f4: [dim=2,faces=(e24,e46,`e62),val=`face]
} ;;
```

# Surface subdivision



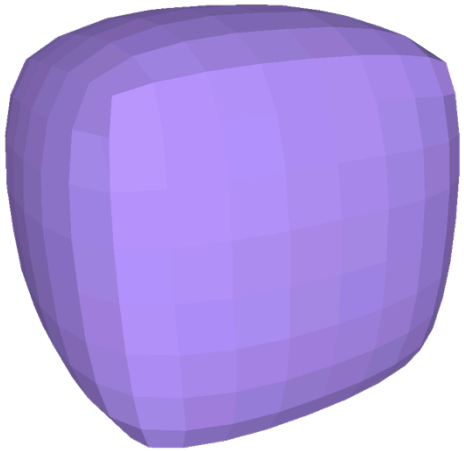
# Surface subdivision: Butterfly interpolating mask

```
patch insert_vertex[genNb] = {
  ~f1:[dim=2,faces=(e,e23,e13)]
  ~f2:[dim=2,faces=(e,e24,e14)]
  ~ff1:[dim=2,faces=(e13,e35,e15)]
  ~ff2:[dim=2,faces=(e23,e36,e26)]
  ~ff3:[dim=2,faces=(e14,e17,e47)]
  ~ff4:[dim=2,faces=(e24,e28,e48)]

  ~v2 < ~e23 > ~v3 ~e15 > ~v5 < ~e35
  ~v1 < ~e13 > ~v3 ~e26 > ~v6 < ~e36
  ~v2 < ~e24 > ~v4 ~e17 > ~v7 < ~e47
  ~v1 < ~e14 > ~v4 ~e28 > ~v8 < ~e48

  ~v1 < e > ~v2 =>
    `e1:[dim=1,faces=…,val=`edge]
    `v:[dim=0,cofaces=…,
        val=f(v1,v2,v3,v4,v5,v6,v7,v8)]
    `e2:[dim=1,faces=…,val=`edge]
} ;;
```

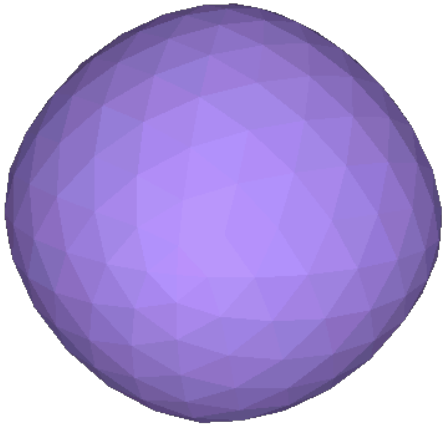


# Surface subdivision (from a cube)

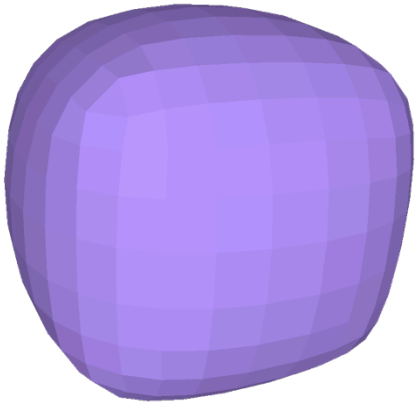
Butterfly



Kobbelt

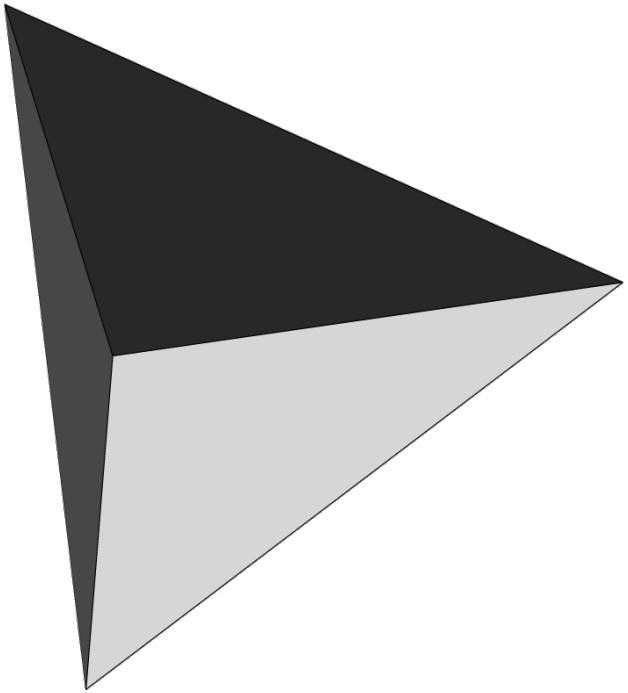


Loop

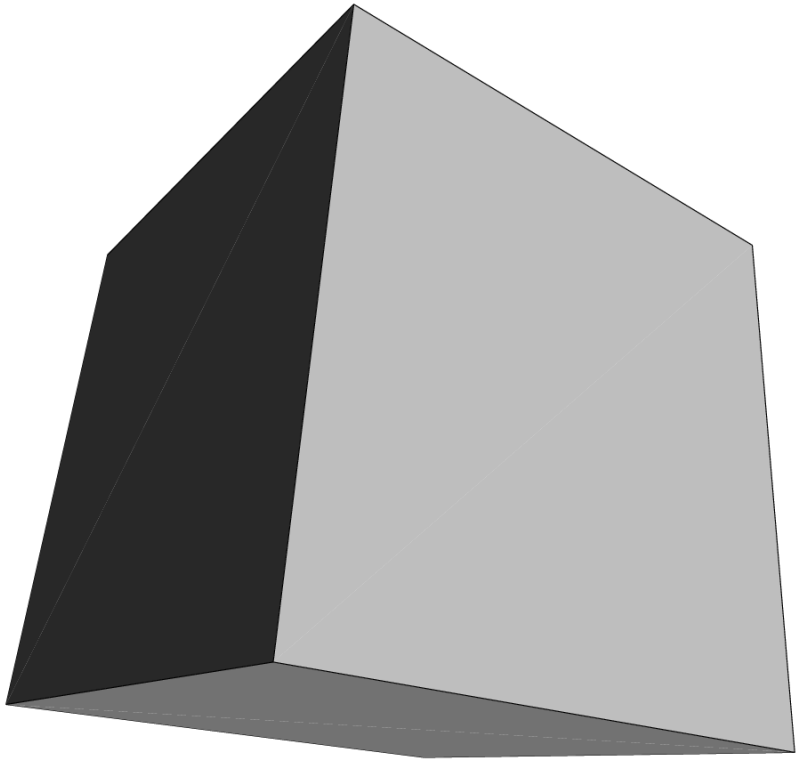


Doo-Sabin
(bevelled edges)

# Fractal construction by carving



Sierpinsky sponge (4 steps)



Menger sponge (2 steps)

# Modeling of (bio)physical systems



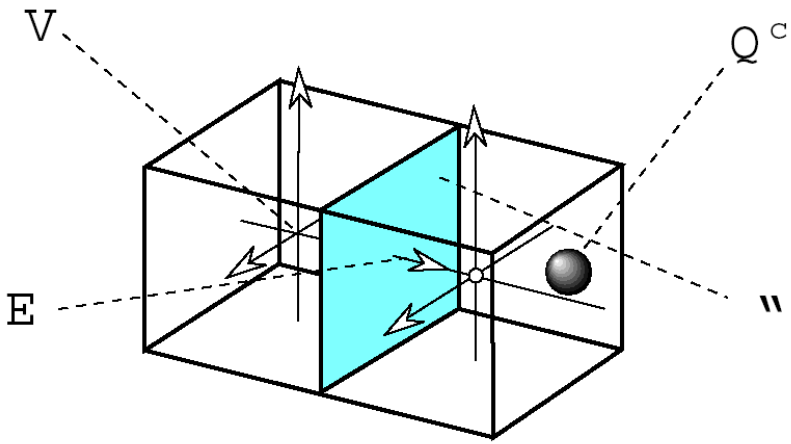
- …
- Meristem growth
- Various models of Phage λ
- Sperm crawling
- Neurulation
- Prototyping a
  « synthetic multicellular bacteria »
- …

Example of electrostatic Gauss law [**Tonti** 74]

- Electric charge content ρ : dimension 3
- Electric flux $\Phi$ : dimension 2
- Law available on a arbitrary complex domain

$$\phi = \oiint w \cdot dS = \frac{Q^c}{\varepsilon_0} = \iiint_{(V)} \frac{\rho}{\varepsilon_0} d\tau$$

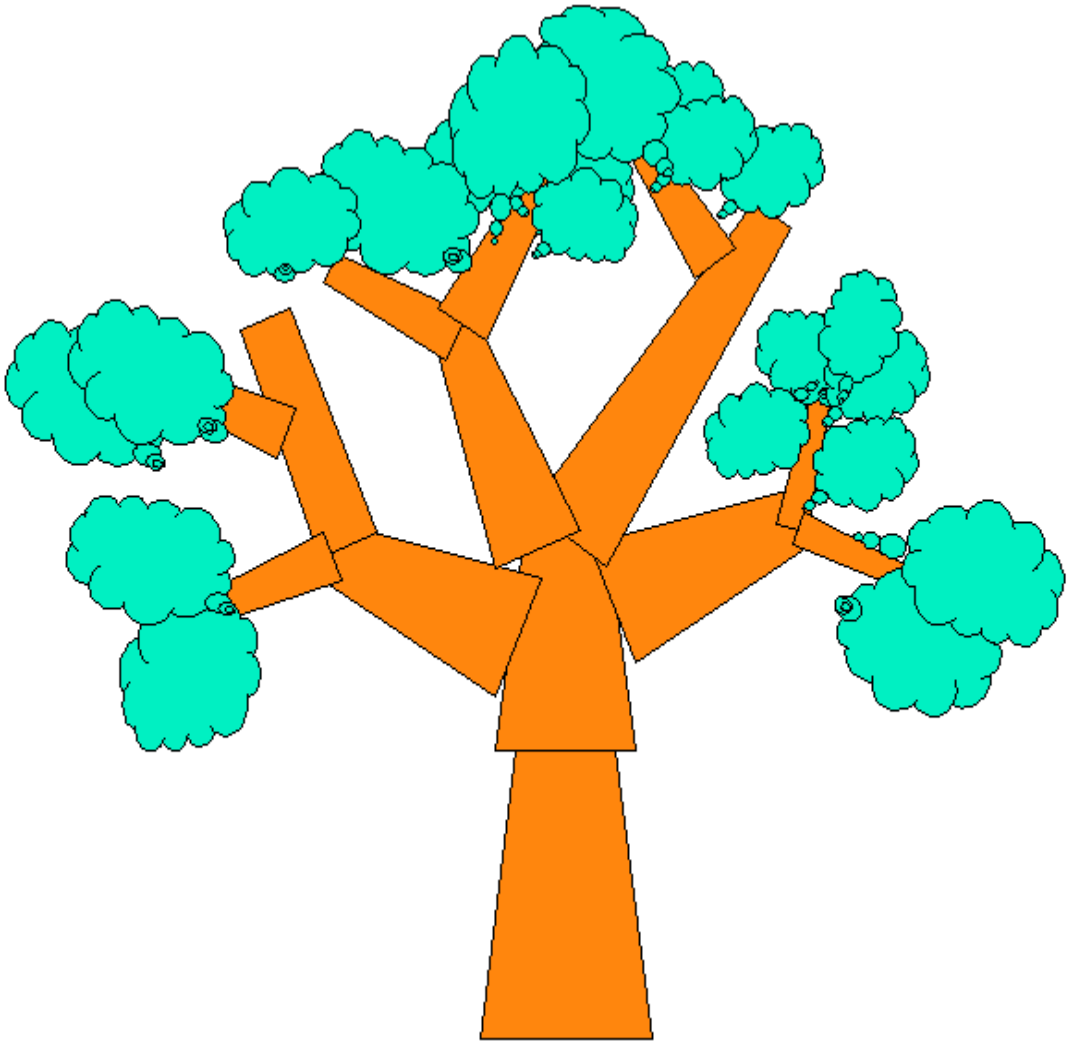
*electric field in space*:
- V: electric potential (dim 0)
- E: voltage (dim 1)
- w: electric flux (dim 2)
- Qc: electric charge (dim 3)



# Complexity of Biological Processes

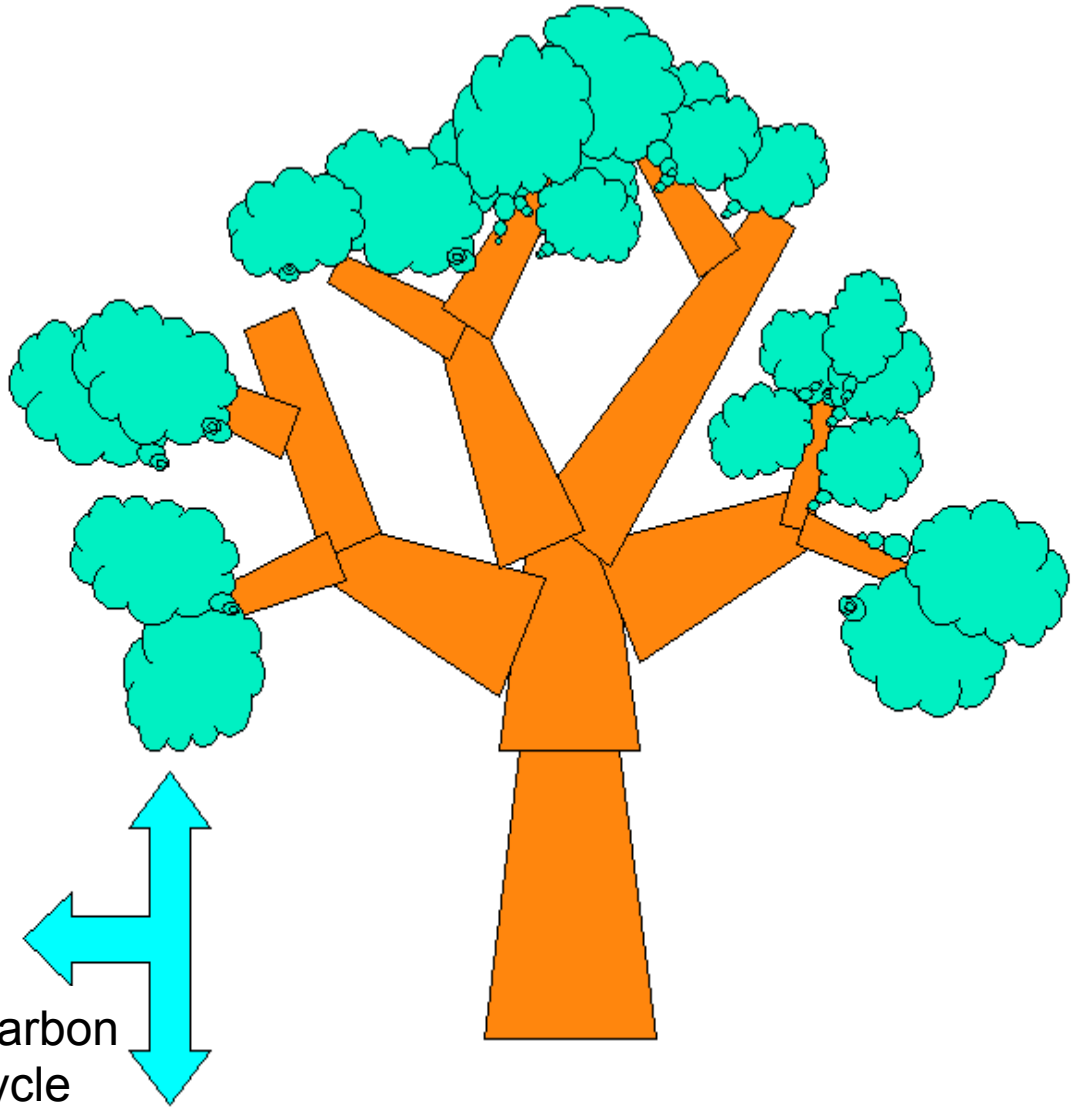
Describe a tree

# Complexity of Biological Processes



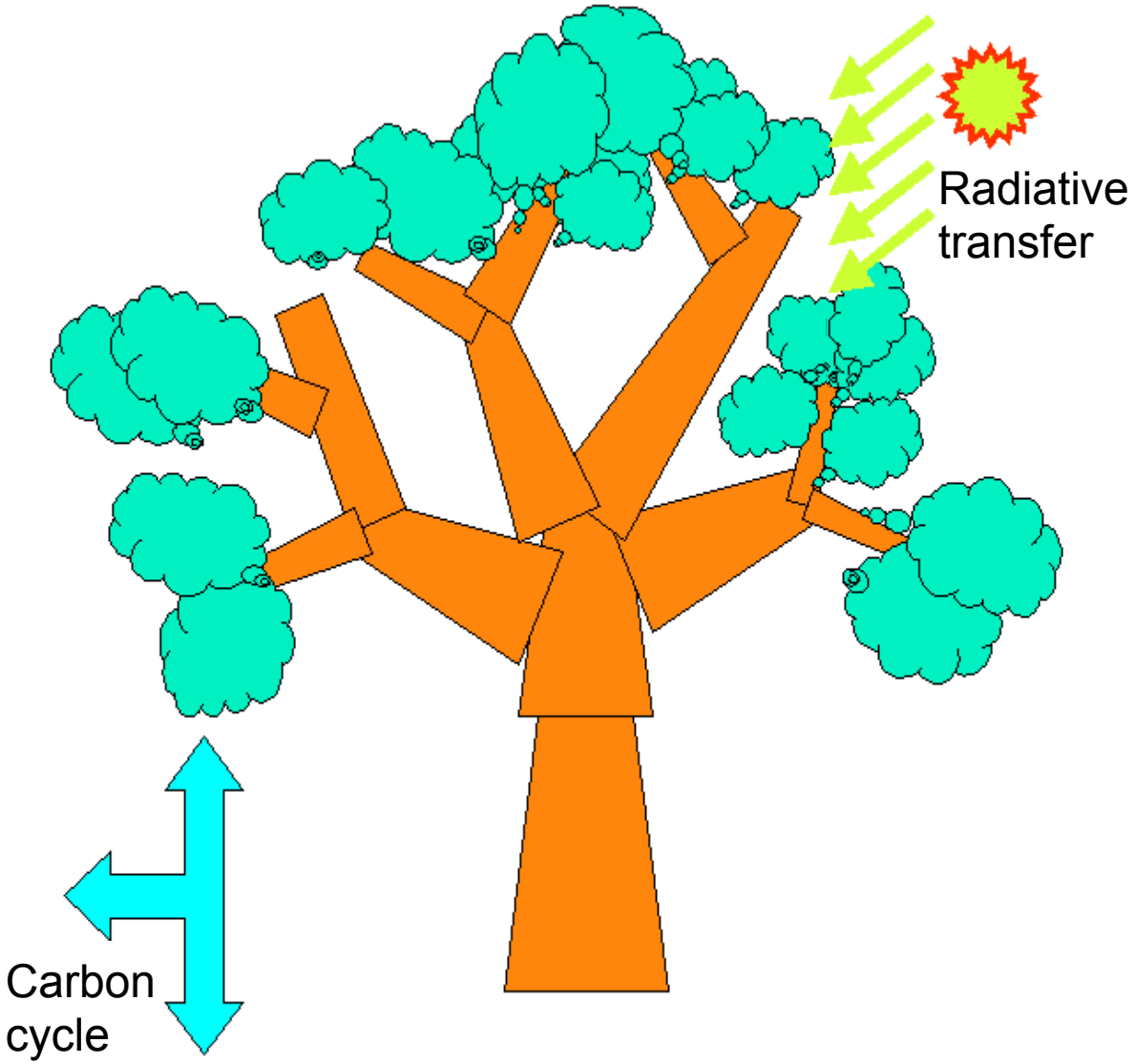
Describe a tree

# Complexity of Biological Processes



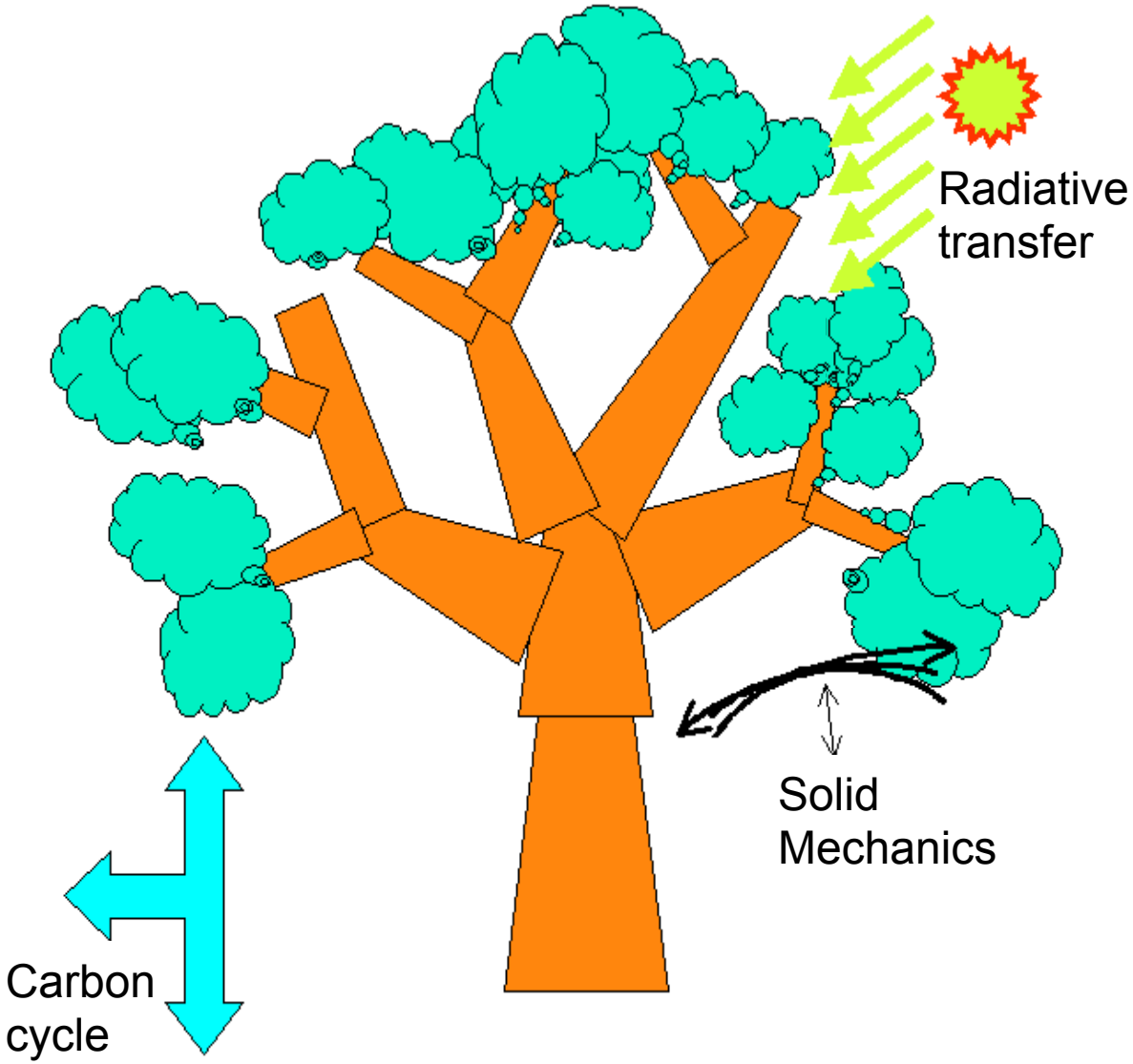

Describe a tree

# Complexity of Biological Processes



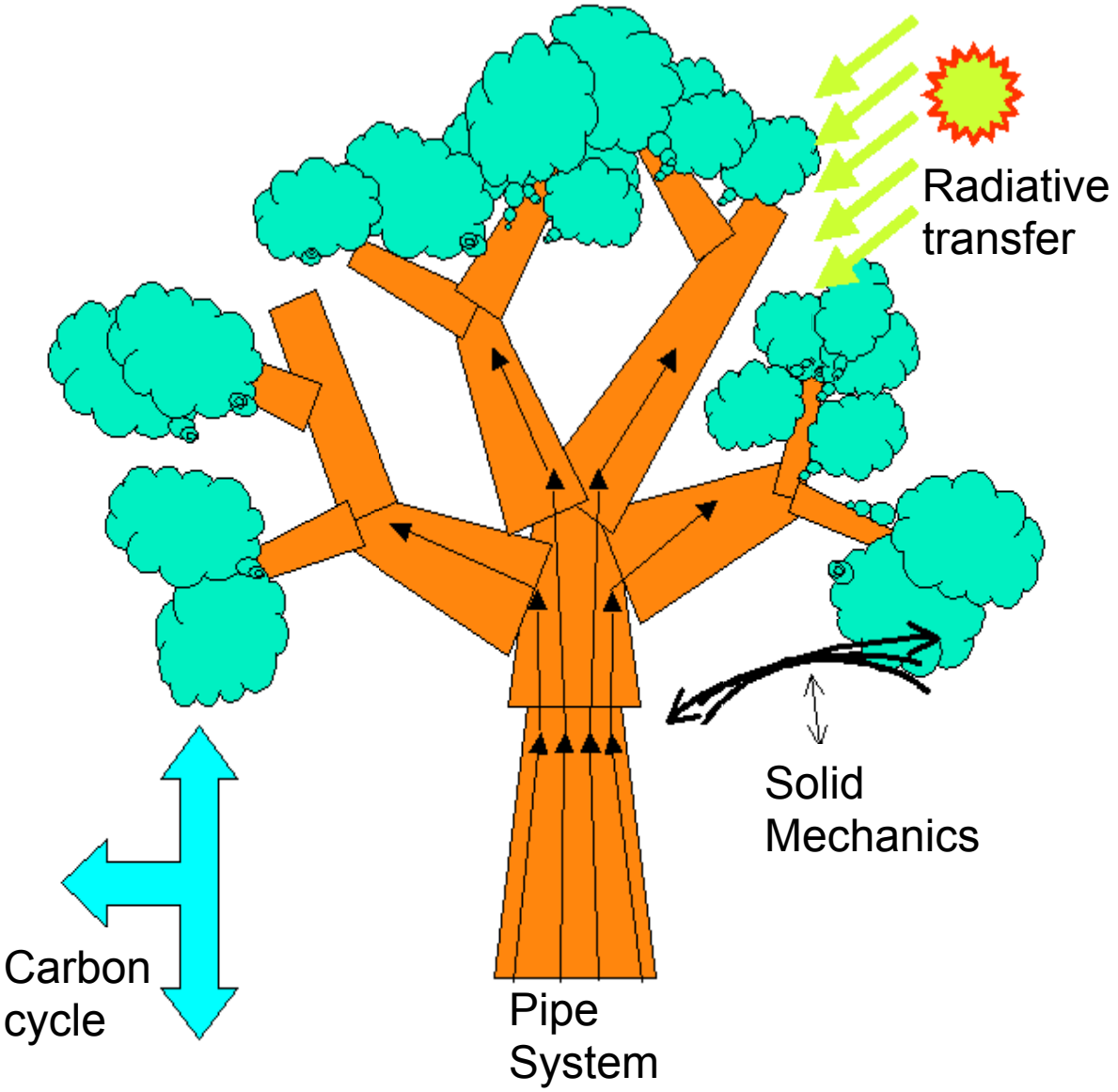

Describe a tree

# Complexity of Biological Processes



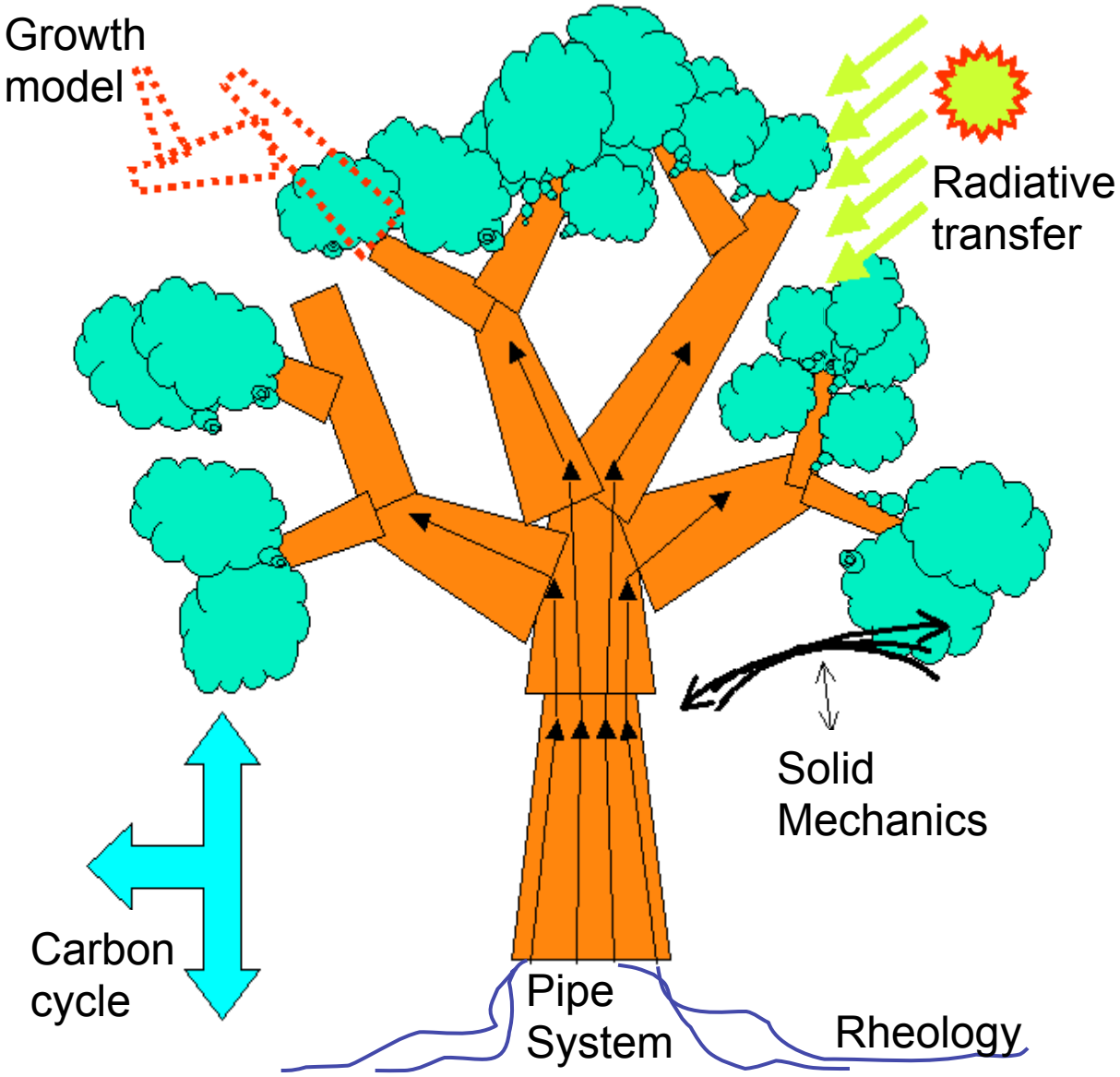

Describe a tree

# Complexity of Biological Processes



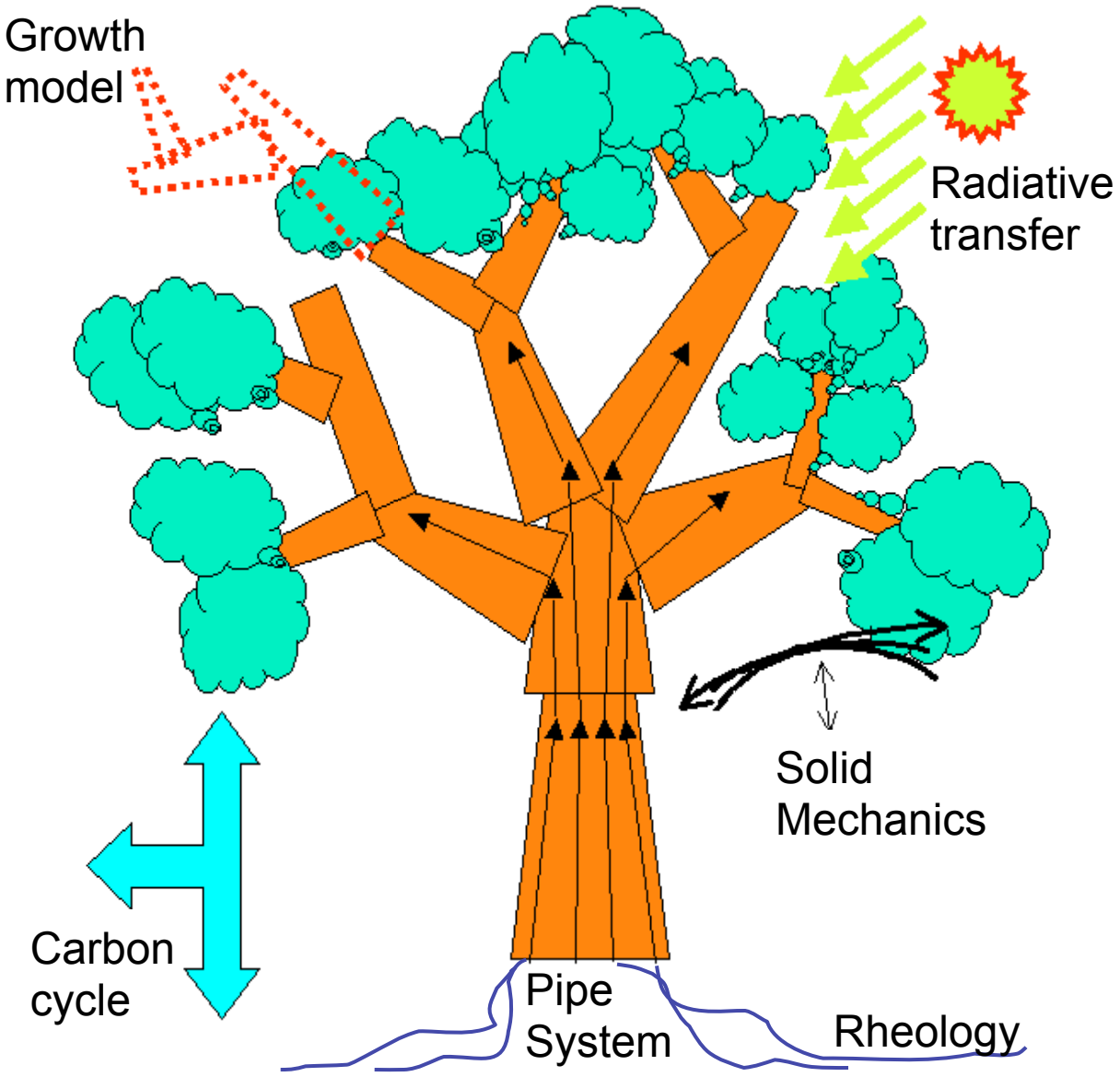

Describe a tree

# Complexity of Biological Processes




Describe a tree

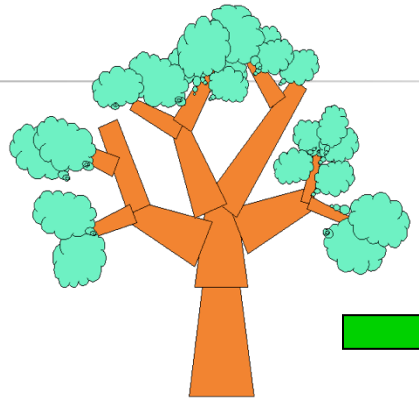
# Complexity of Biological Processes




## Describe a tree

One has to describe

- The physical structure (organs, tissue, membranes, compartments… and their relationships: neighborhood, inclusion…)
- Its state (chemical, electrical…)
- The dynamics
- The geometry and its evolution (position and evolution of each sub-system, shape…)
- Operational aspects of the computations (data structures, distribution/parallelism…)
- …



**Self-organizing tree models for image synthesis.** Wojciech Palubicki1, Kipp Horel1, Steven Longay1, Adam Runions1, Brendan Lane1, Radomir Mech2, and Przemyslaw Prusinkiewicz1. *ACM Transactions on Graphics* 28(3), 58:1-10, 2009.

**Is it really that complex?**
**It is even more complex than that!**
**A short bibliography on the subject :**


**•BioChemistry** (Auxin Transport)
Przemyslaw Prusinkiewicz, Scott Crawford, Richard S. Smith, Karin Ljung, Tom Bennett, Veronica Ongaro, and Ottoline Leyser.
Control of bud activation by an auxin transport switch. *Proceedings of the National Academy of Sciences* 106 (41), pp. 17431-17436, 2009.

**•Image Synthetesis** (rendering)
Wojciech Palubicki, Kipp Horel, Steven Longay, Adam Runions, Brendan Lane, Radomir Mech, and Przemyslaw Prusinkiewicz.
Self-organizing tree models for image synthesis. *ACM Transactions on Graphics* 28(3), 58:1-10, 2009.

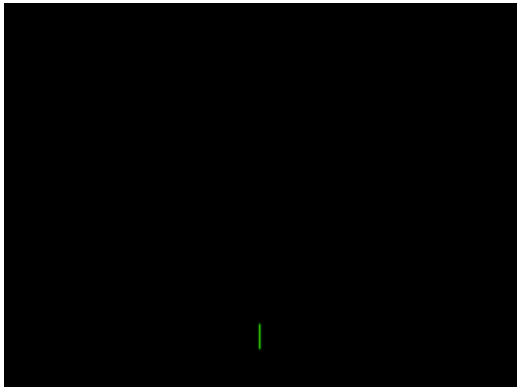
**•Phyllotaxis**
Emmanuelle M. Bayer, Richard S. Smith, Therese Mandel, Naomi Nakayama, Michael Sauer, Przemyslaw Prusinkiewicz, and Cris Kuhlemeier.
Integration of transport-based models for phyllotaxis and midvein formation. *Genes & Development* 23(3), pp. 373-384, 2009.

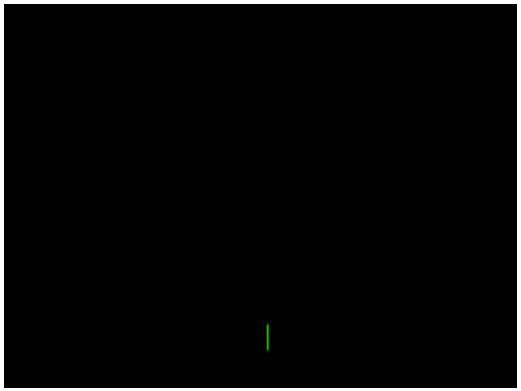
**•BioMechanics**
Evylene Costes, Colin Smith, Michael Renton, Yann Guédon, Przemyslaw Prusinkiewicz, and Christophe Godin.
MAppleT: simulation of apple tree development using mixed stochastic and biomechanical models. *Functional Plant Biology* 35(10), pp. 936-950, 2008.

**•Architecture**
Gerardo Lopez, Romeo R. Favreau, Colin Smith, Evelyne Costes, Przemyslaw Prusinkiewicz, and Theodore M. DeJong.
Integrating simulation of architectural development and source-sink behaviour of peach trees by incorporating Markov chains and physiological organ function submodels into L-PEACH. *Functional Plant Biology* 35(10), pp. 761-771, 2008.

**•Radiative Transfer** (Light)
Mikolaj Cieslak, Christiane Lemieux, Jim Hanan, and Przemyslaw Prusinkiewicz. Quasi-Monte Carlo simulation of the light environment of plants.*Functional Plant Biology* 35(10), pp. 837-849, 2008.

# A good example of **declarative** formalism: Lindenmayer systems





- The structure of a tree can be coded by a string of parenthetised symbols
- A symbol is an elementary part of the plant
- The symbol between [ and ] represents a sub-tree
- Additional conventions are used to represent main axis, orientation, depth, etc.
- A rule

  $s_0 \rightarrow s_1\ s_2\ s_3\ \ldots$

  represents the evolution of $s_0$

P. Prusinkiewicz

# Diffusion and reaction in a linear *growing medium*

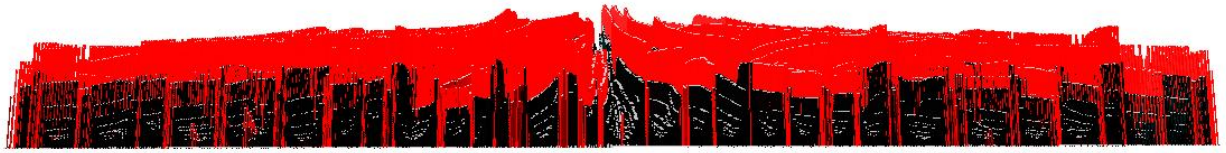
M. Hammel and P. Prusinkiewicz (1996)

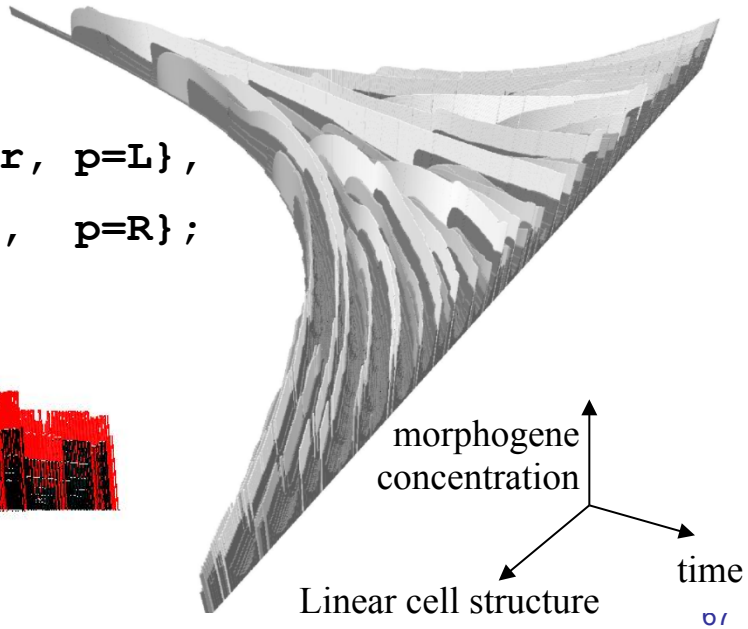
The following rules state that a differentiated cell (heterocyst) returns to a vegetative state if the concentration of the activator is too low. In addition, if the cell is large enough, it continues to grow.

```
e / (D(e) & (e.a < thr) | (e.x >= shorter*gr))
=> {type ="C", a=e.a/gr, h=e.h/gr, x=e.x*gr, p=e.p};
```

The following rule specifies when a cell with a left polarity divides. Only vegetative cells can divide (hence the predicate C in the rule guard) and it must be large enough. The volume of the two daughter cells remains the same, so there is no variation in the concentration.

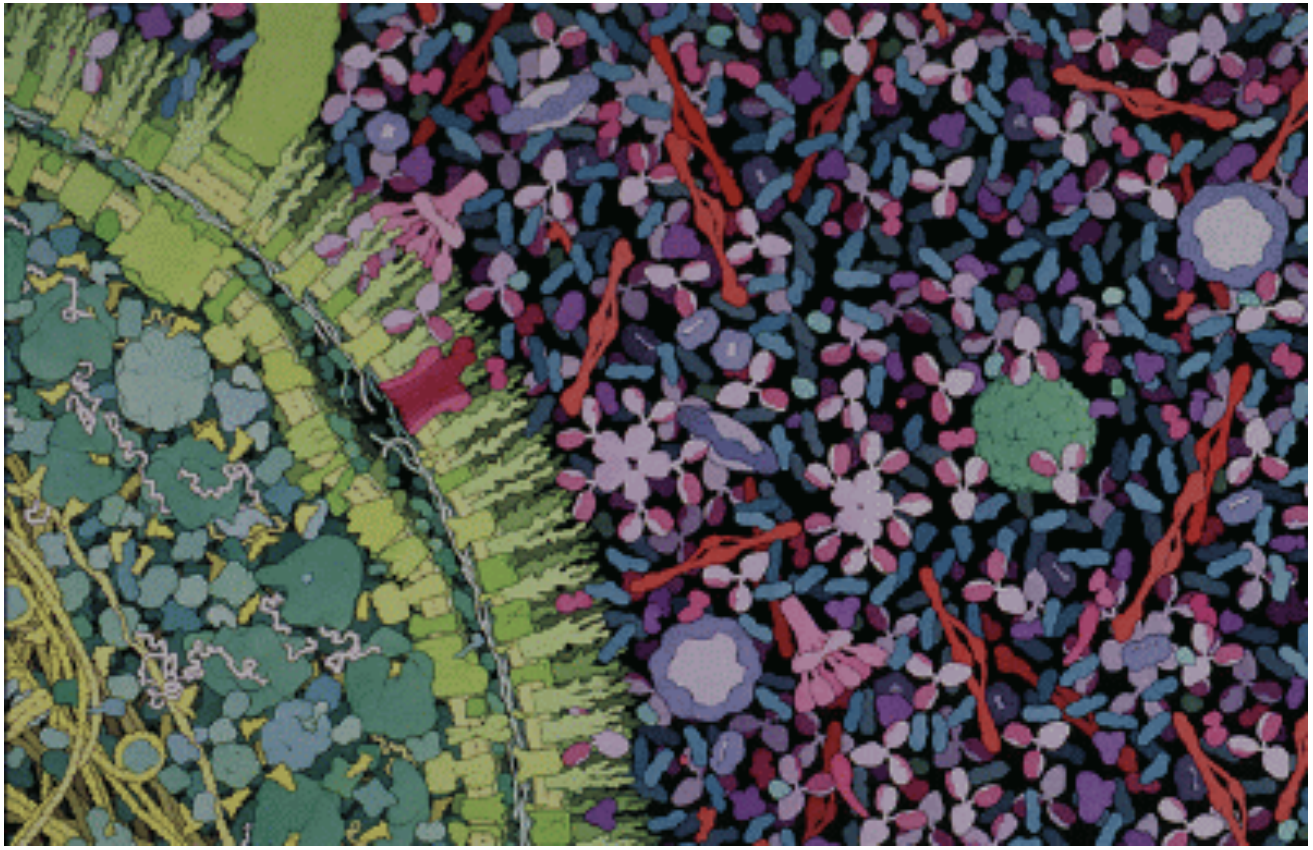
```
e / (C(e) & (e.x >= lm) & (e.p == L))
=> {type="C", a=e.a, h=e.h, x=e.x*shorter, p=L},
   {type="C", a=e.a, h=e.h, x=e.x*longer,  p=R};
```





# How to extend to arbitrary spatial structure?

- Anabaena was « easy » because of the linear uniform structure
- How to handle the complex spatial structure of a cell?



David S. Goodsell

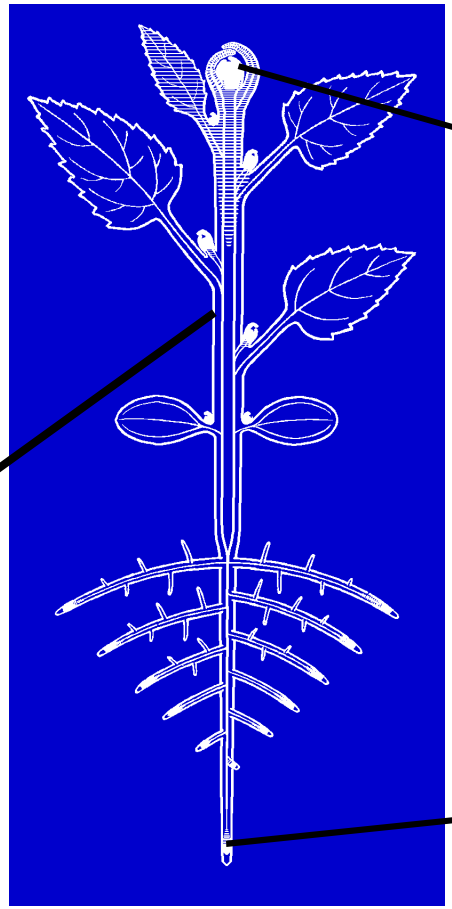
# The Growth of a Meristem

## [PNAS 103(5), 1627-1632, 2006]

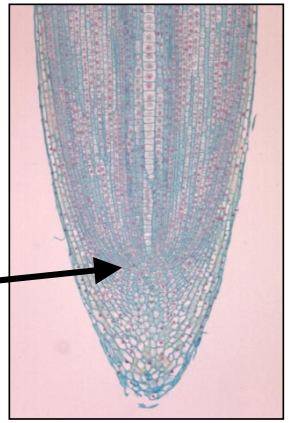
Pierre Barbier de Reuille
Mikaël Lucas
Jan Traas
Christophe Godin
CIRAD/INRA/INRIA

Organs positionning at the shoot apex

Cambium

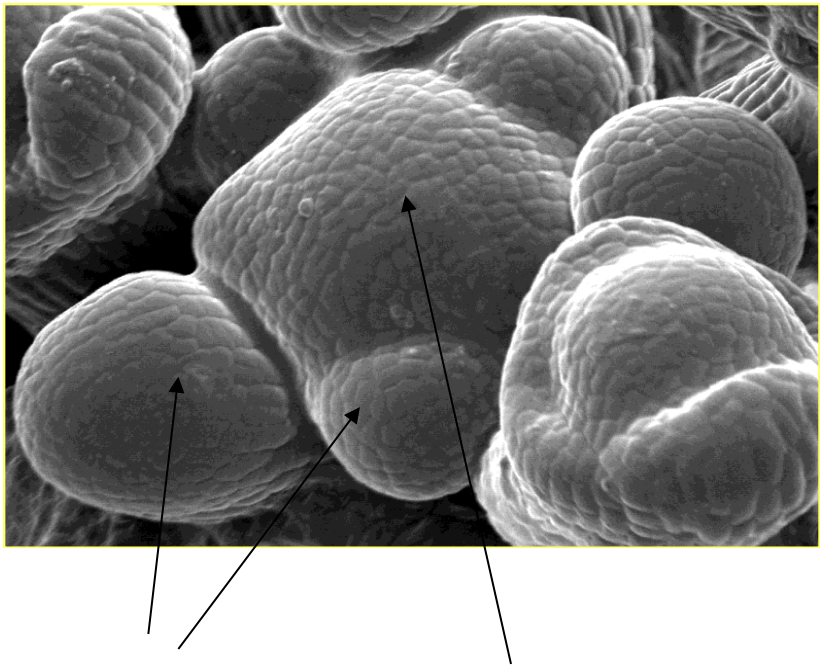


Shoot apical meristem



Root apical meristem

# A shoot apical meristem



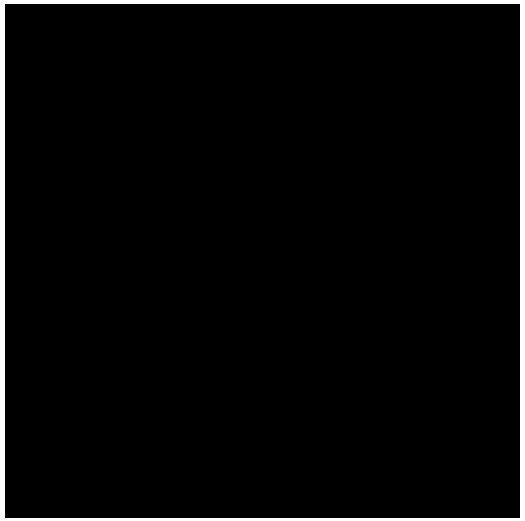
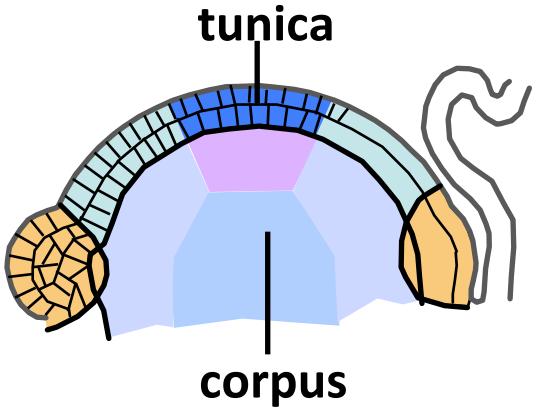
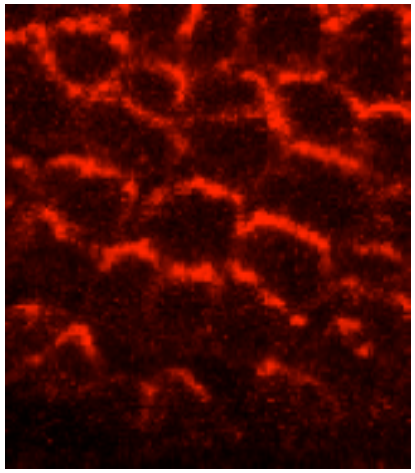



Image sequence showing cell division patterns via membrane-bound PIN1, in Shoot Apical Meristem (SAM), nearby floral meristems, and the boundaries between them (M. Heisler). http://computableplant.ics.uci.edu/ (E. Mjolness)

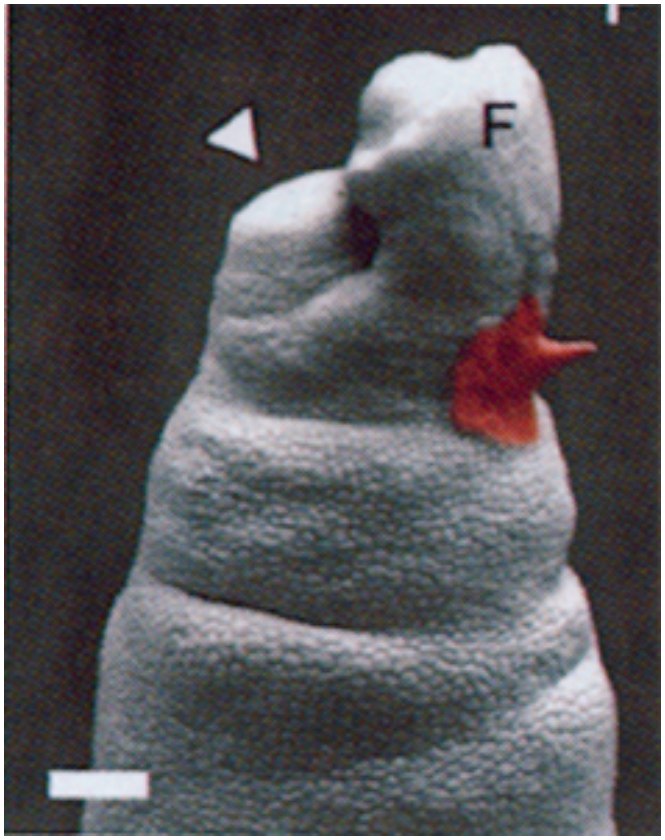


# Active transport of auxine

wild type

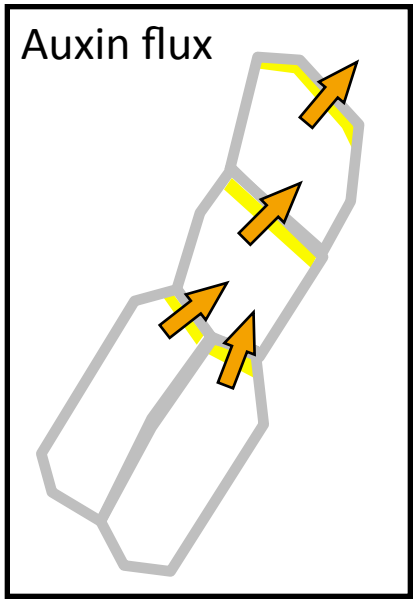


Immunolabelling of
PIN-FORMED1 protein

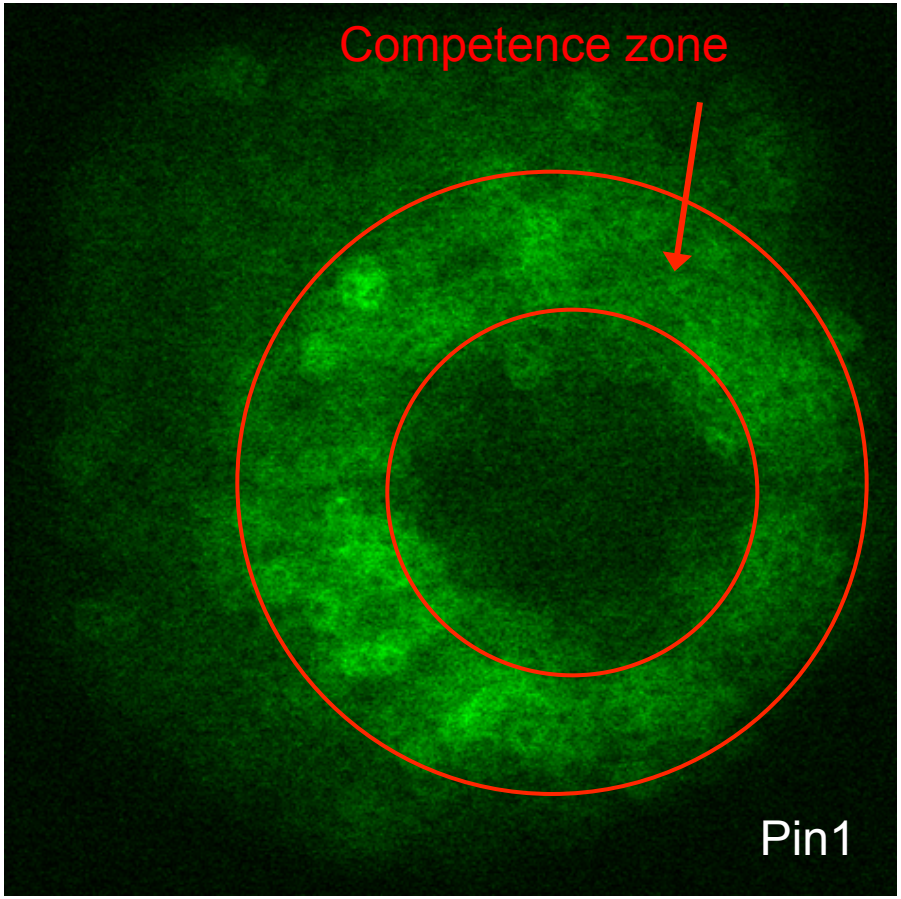



high concentration of
auxine induces organ initiation

pin-1 mutant



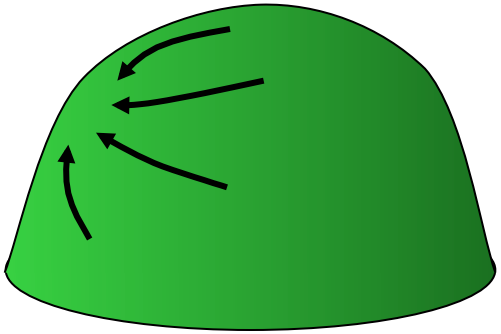
# Genetic labelling

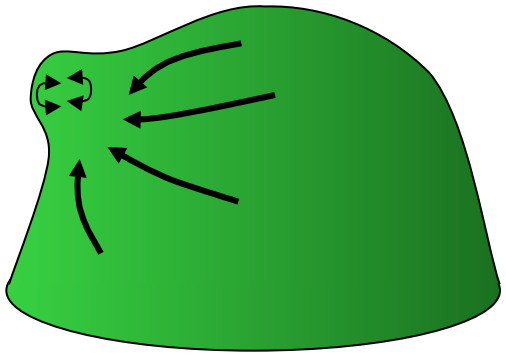

Images : Vernoux & Traas

ANT::GFP

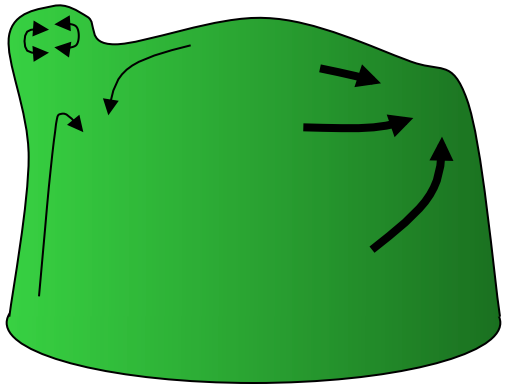
# $(DS)^2$



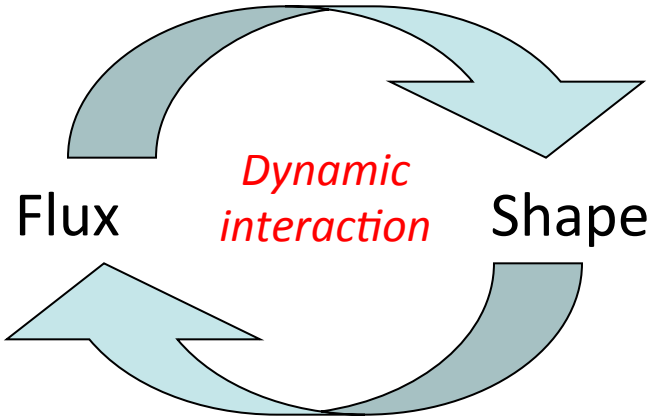
flux…



changes form…

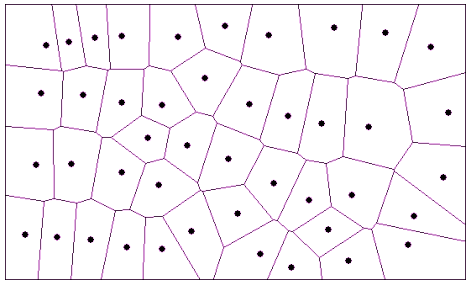


which changes flux…

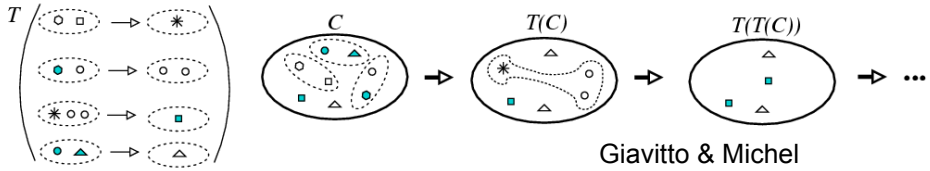


# Virtual meristem

1 – Meristem representation

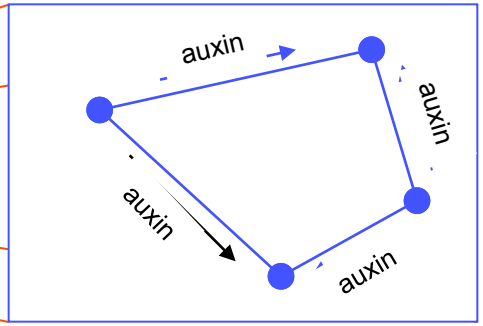


2 – Growth model $(DS)^2$

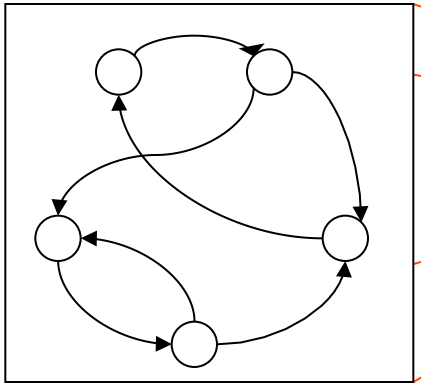


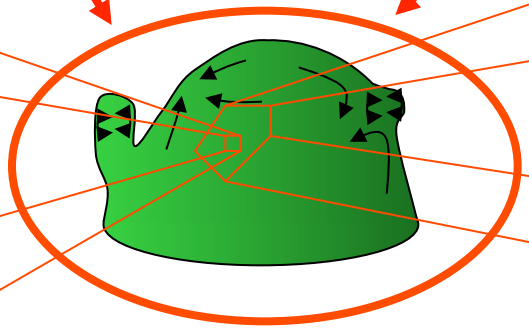

Giavitto & Michel

3 – Transport model

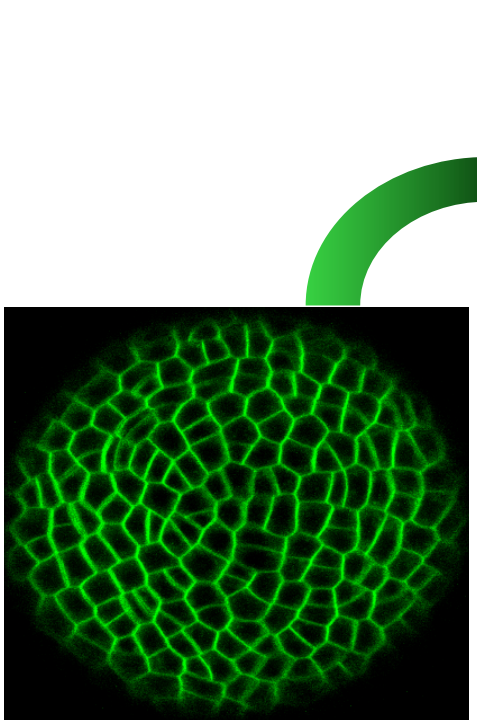



4 – Cell model

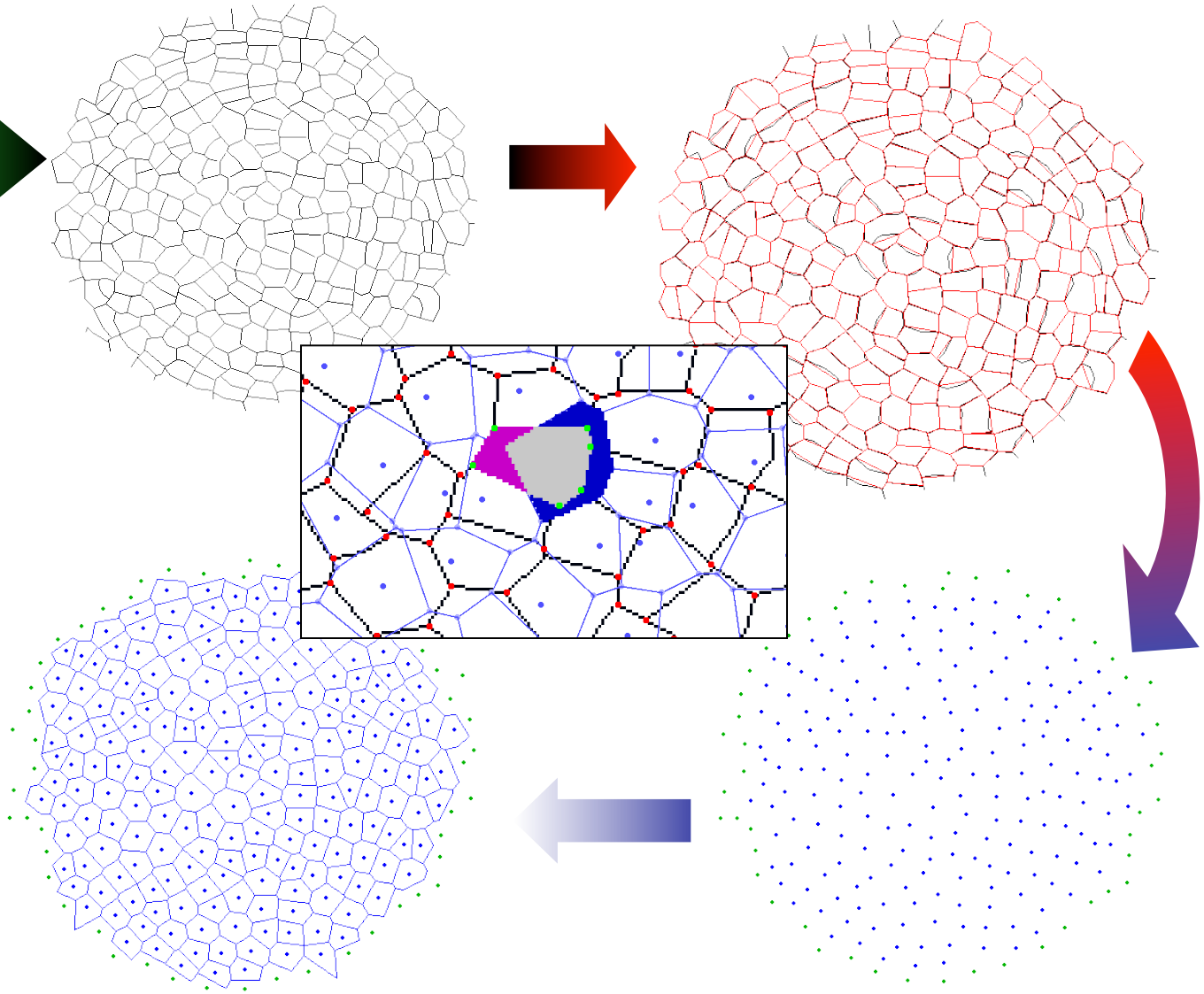




# Meristem representation

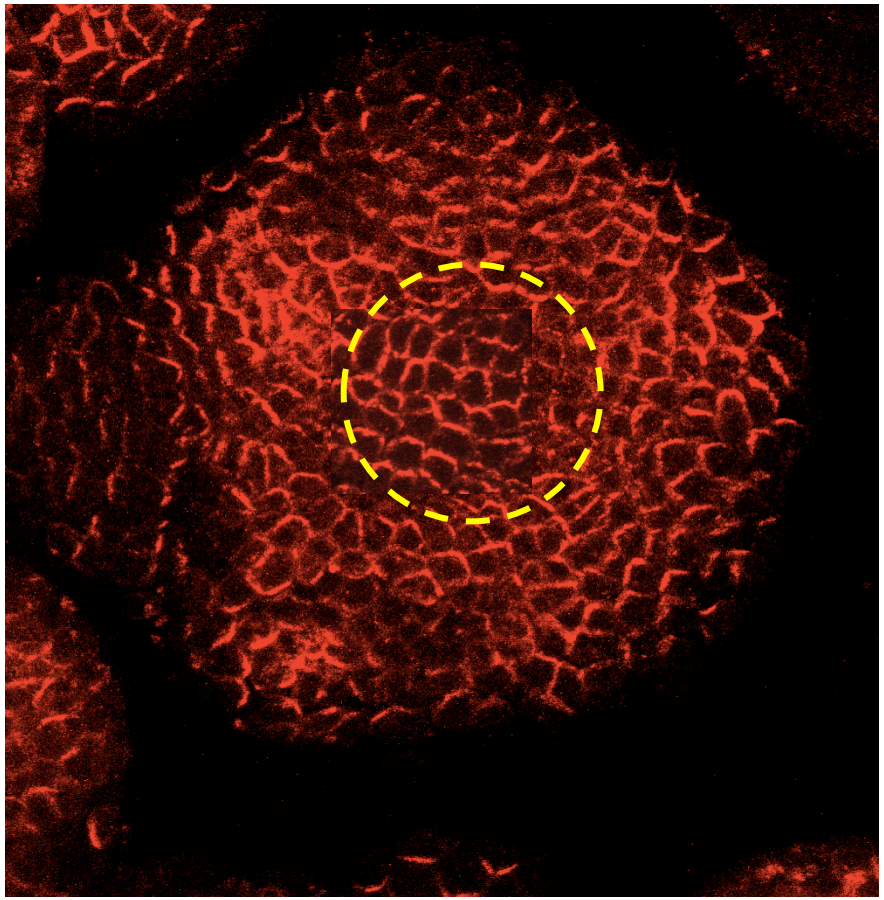


(Honda 1978, 2004)

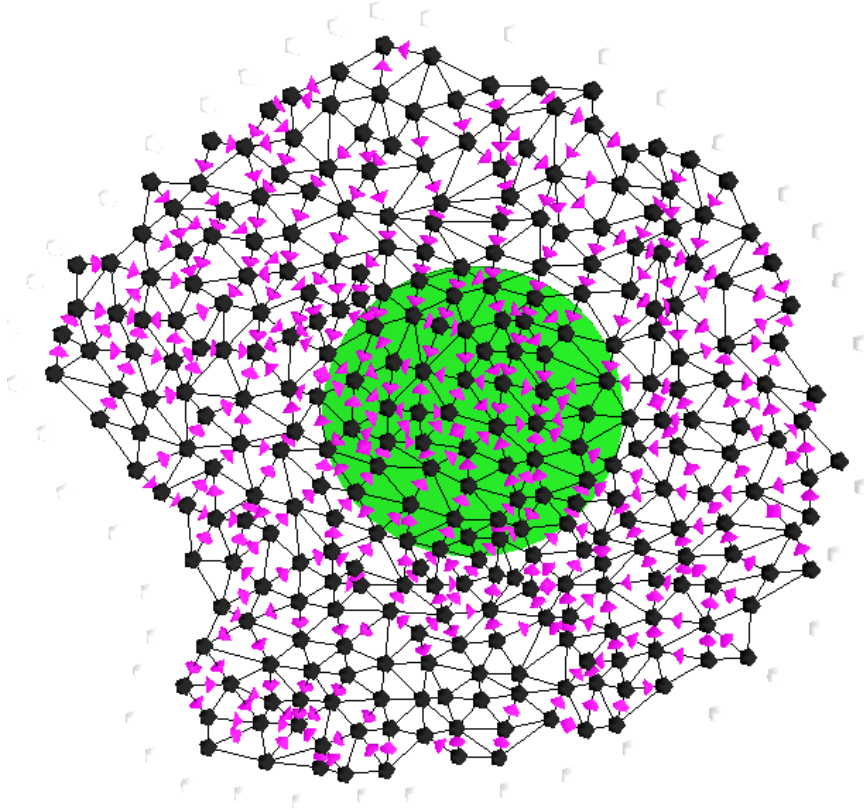


(Barbier de Reuille et al. 2003)

# Meristem representation



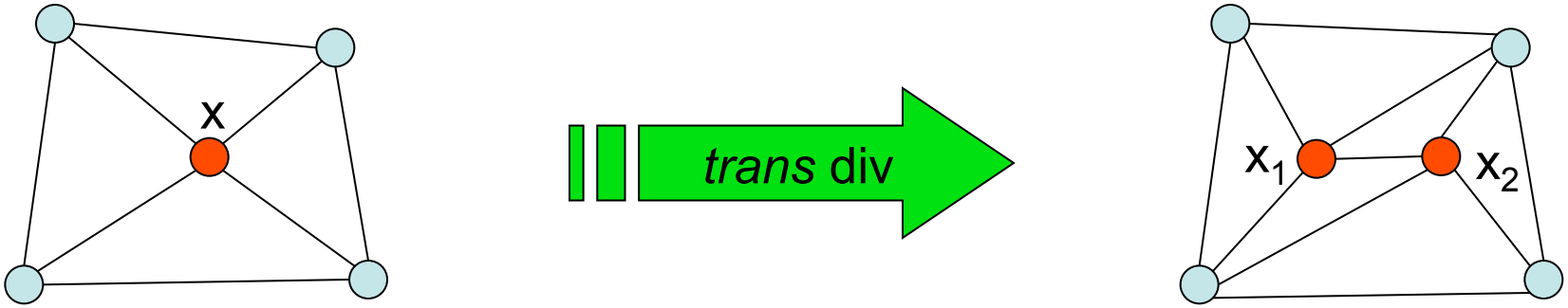
Original



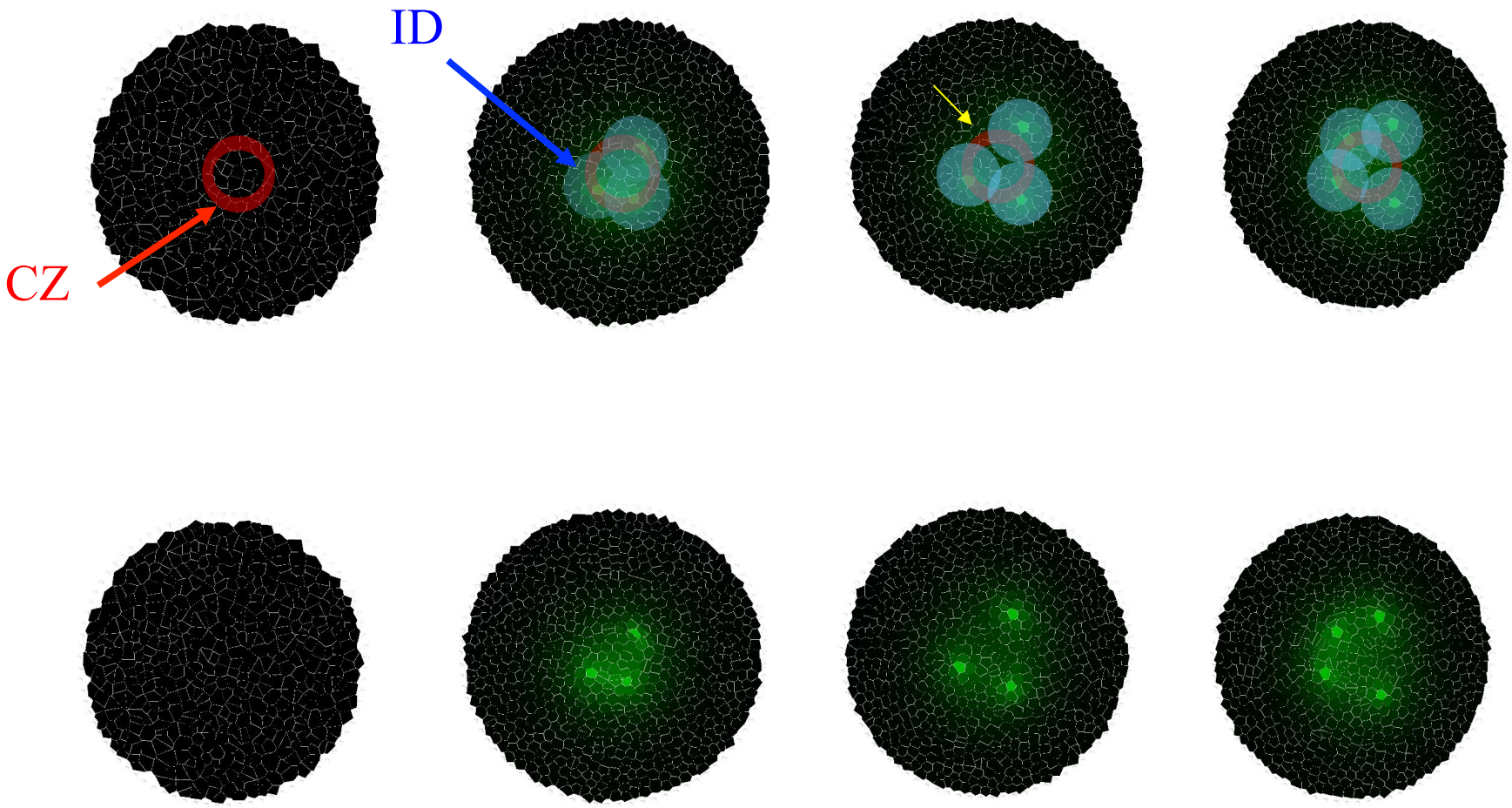
Digitalized

# MGS – dynamic structure rules

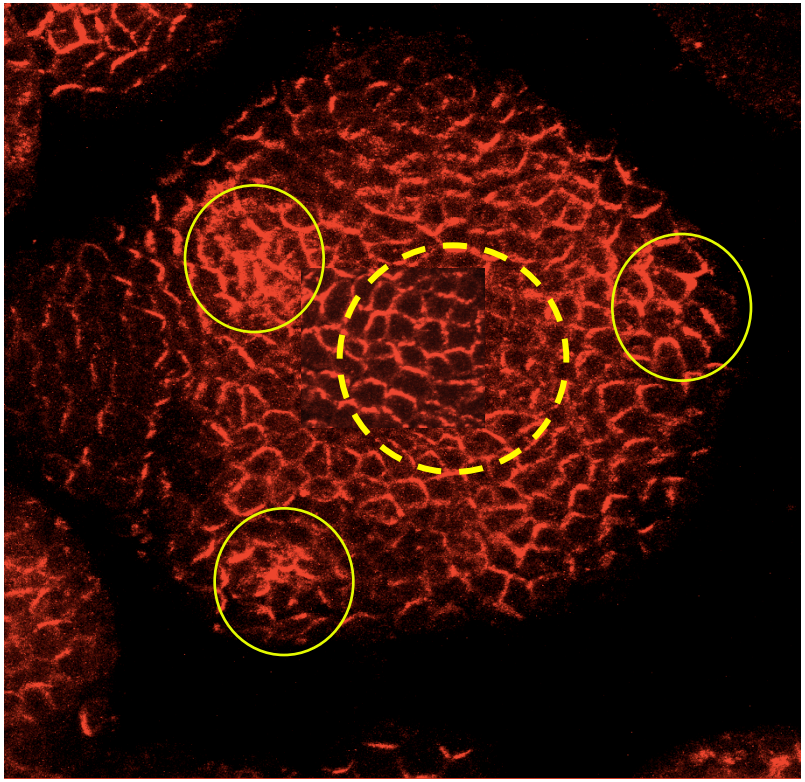
$$trans\ \mathrm{div} = \{x / dividing(x) \Rightarrow child(x,1), child(x,2)\}$$

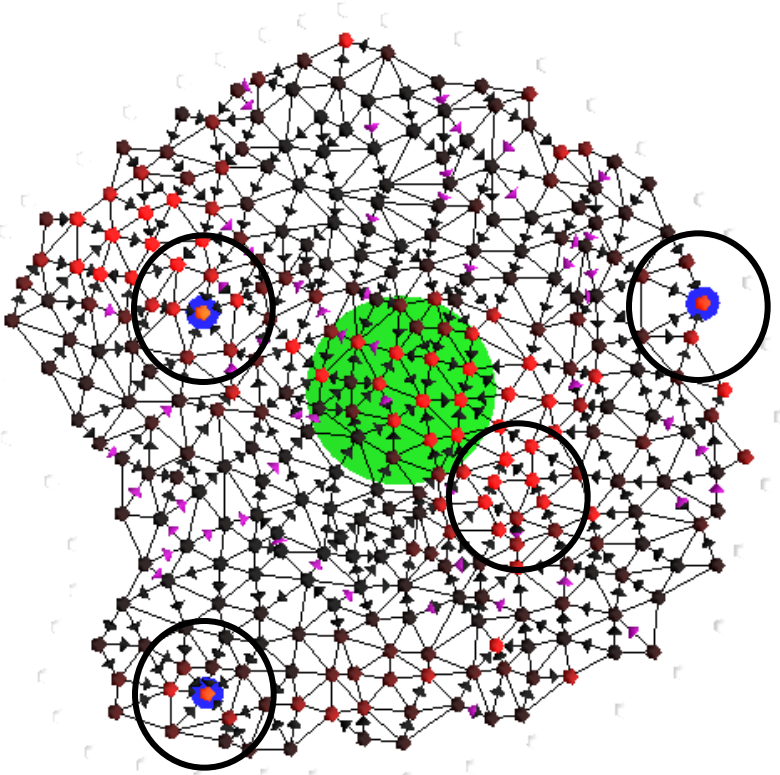


# Model 3 - “Inhibitor fields” and diffusion



# Simulation results





# Model 4 : Active pumping of auxin

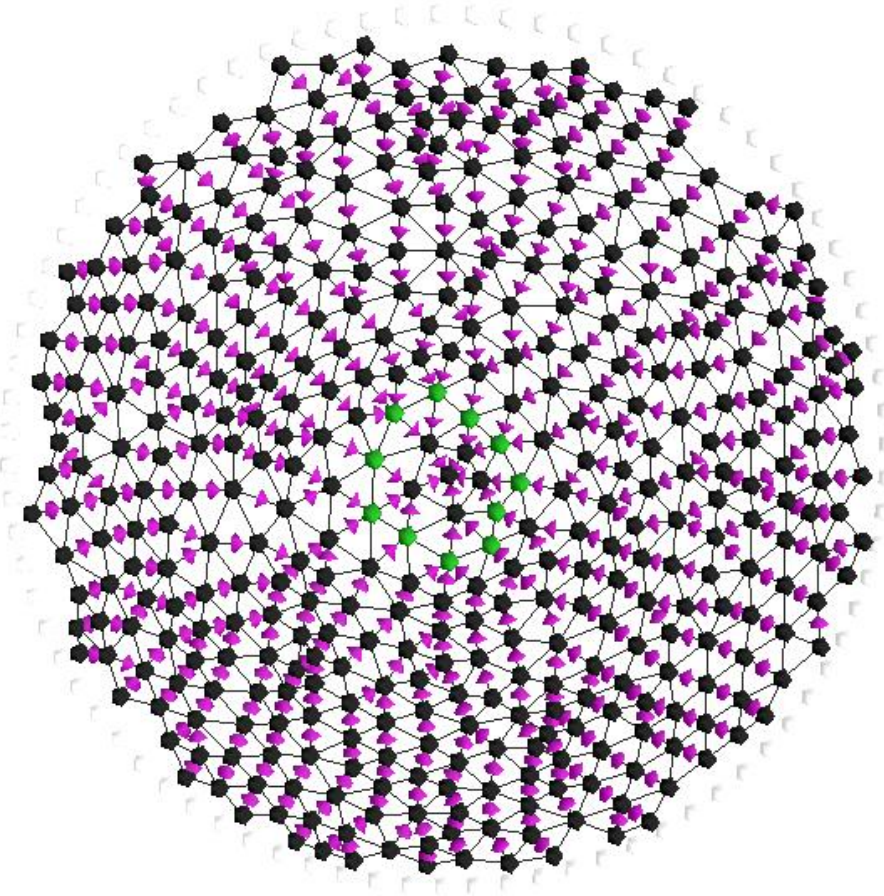
- *Cell internal state and processes =>*
  - capacity of division,
  - spring relaxed length,
  - primordium/center,
  - **concentration of auxin (inhibitor),**
  - **saturation,**
  - **auxin degradation / evacuation**
  - **promotion to primordium**
  - **“pump magnetism”**

- *Movement =>*
  - due to cell growth

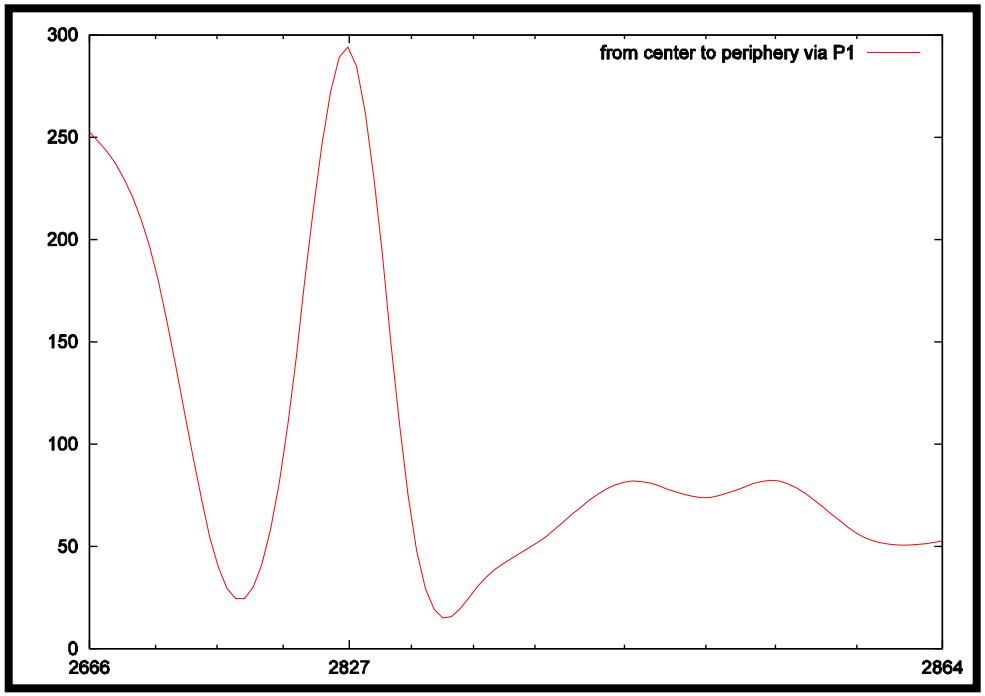
- *Growth =>*
  - increase of spring relaxed length

- *Division =>*
  - when size > threshold

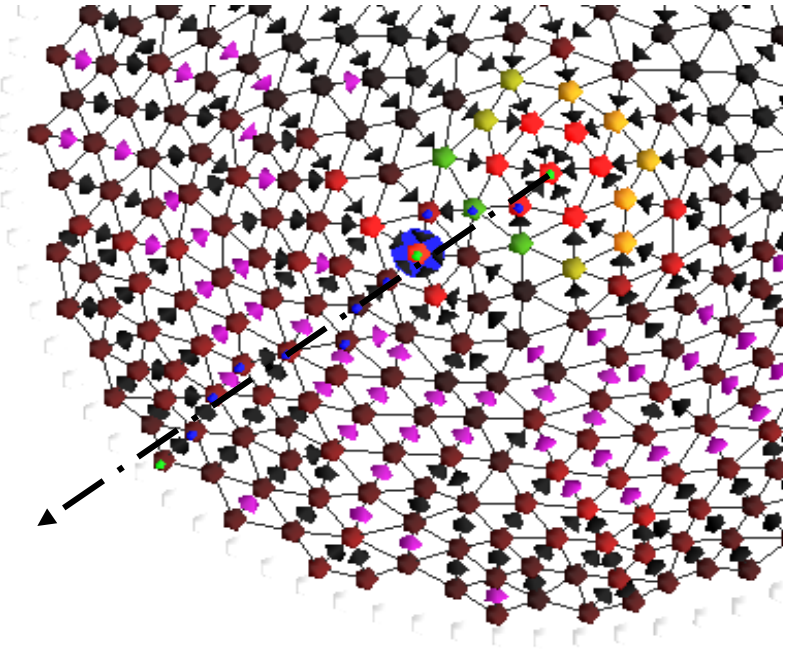
- *Cell interaction =>*
  - **Passive diffusion of auxin,**
  - **active pumping of auxin**



# Primordium local inhibition



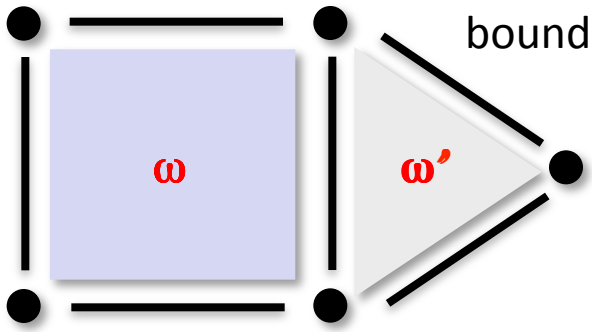

Auxin level



# Transformations et opérateurs différentiels (discrets)

# Example: (some special) Transformations as Cochains

- The Boundary Operator $\partial$
  - Starting point of the elaboration of a discrete diff. calculus
  - *Transport* of data *from cells to their faces*




  - Cochains notation

    The boundary operator is a cochain

$$\partial = \sum_{\sigma \in \mathcal{K}} \partial_\sigma . \sigma \quad \text{with} \quad \forall \sigma \in \mathcal{K},\ \partial_\sigma(g) = \sum_{\tau < \sigma} o_{\sigma\tau}(g).\tau$$

  - MGS notation

```
trans Boundary      = {
  x => CofacesFold(fun y acc -> o~y^x(y) +G acc,0G,^x) }
```

# Transformation as Cochains

- Derivative Operator $\mathbf{d}$
  - Defined w.r.t. the discrete Stockes' Theorem

$$[\mathbf{d}T, c] = [T, \partial c]$$

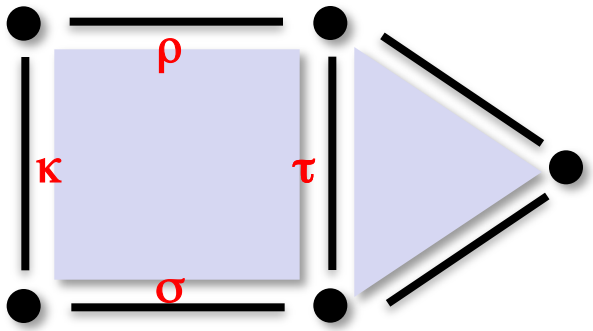
  - Cochains Notation

    One can show that the derivative verifies

$$\mathbf{d} = \sum_{\tau \in \mathcal{K}} \mathbf{d}_\tau . \tau \quad \text{with} \quad \forall \tau \in \mathcal{K},\ \mathbf{d}_\tau(f) = \sum_{\tau < \sigma} (f \circ o_{\sigma\tau}) . \sigma$$

  - MGS Notation

    We directly use the Stockes' Theorem




```
let Derivative  T = fun c -> T (Boundary c)
```

# Transformation as Cochains

- Illustrative example : the Laplacian Operator $\Delta$
  - The Laplacian in terms of $\star$ and **d** [Desbrun *et al.*, 2006]

$$\Delta = \delta\mathbf{d} + \mathbf{d}\delta \qquad \text{where } \delta = (-1)^{n(k-1)+1} \star \mathbf{d} \star$$

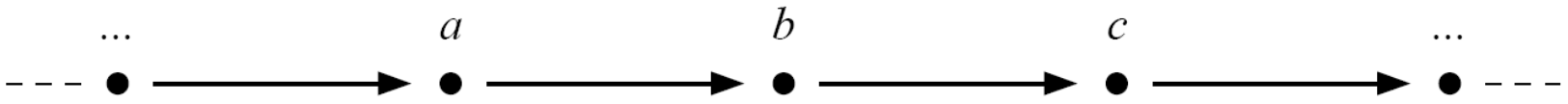
  - MGS notation

    Big assumption: the Hodge star $\star$ is replace by the co-derivative $\mathbf{d}^{co}$
    (= uniform geometry)

```
let Laplacian T =
  let Sg T' c' =
    T'(trans { x => -1**((dim c')*((dim ^x)-1)+1)*x }(c'))
  in
    fun c ->   Derivative(Sg(Derivative^co(T)))(c)
             + Sg(Derivative^co(Derivative(T)))(c)
```

# Transformation as Cochains

- Illustrative example : the Laplacian Operator $\Delta$
  - Corresponding Data Transport (case of dimension 1)
    - Dimension 1: $\Delta = \mathbf{d}^{co}\mathbf{d}$
    - Stockes' Theorem: équivalence with $\partial \circ \partial^{co}$

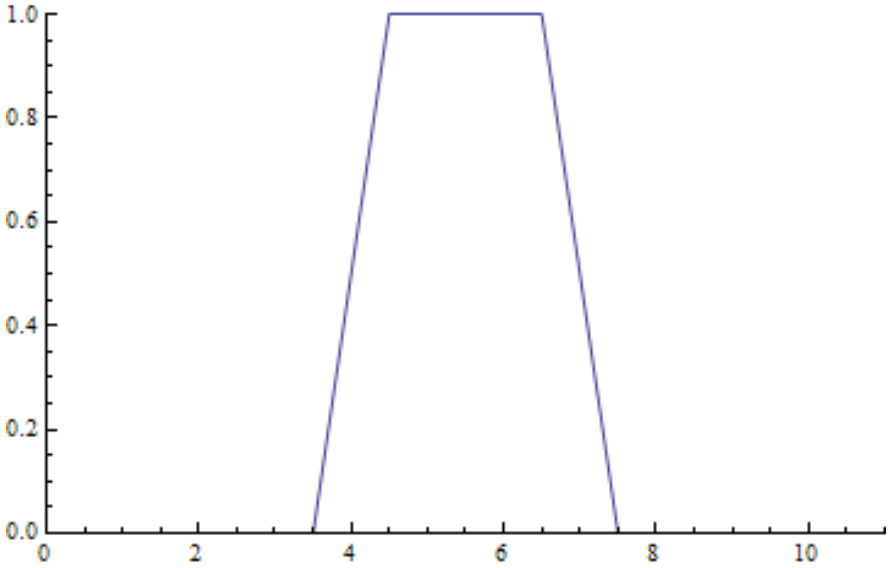


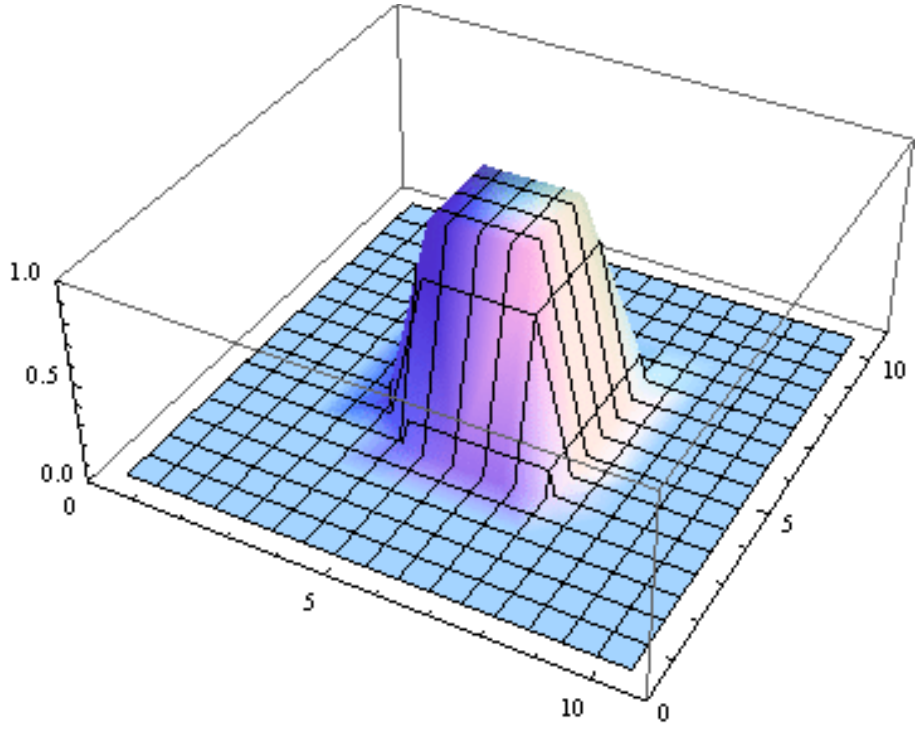
# Transformation as Cochains

- Illustrative example : the Laplacian Operator $\Delta$
  - Simulation of diffusion

$$\frac{\partial u}{\partial t} = D\Delta u$$

```
fun diffusion[D,orient](u) =
  u + D*Laplacian[orient=orient](Id)(u) ;;
```



# Conclusions and Perspectives

# MGS drawbacks and successes



## Success

- Polytypisme is good
- Rule application strategies are good
- Patterns/rules are expressive and usually concise
- Clean semantics

## Shortcomings

- Rules may be heavy (e.g. 100 variables for the fractal sponge)
  *graphical drawing of rules*
  *look for better notations (e.g. path pattern)*
- Efficiency
  *well…*
- Implicit methods (solvers) are hairy
  *use explicit ones*

# Perspectives

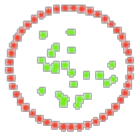


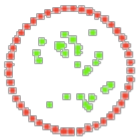
- An intrinsic complexity theory
  e.g., w.r.t. interactions
- A logic of spatial interactions
- Relationships to physics
  a discrete differential calculus (cf. PhysicaD 08)
- Internalizing time
- Implementation
  pattern-matching compilation and optimization, specific abstract combinatorial complex, parallelism
- Non standard applications
  e.g., in knowledge representation
  or in music analysis (Louis Bigo talk)

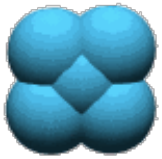
# Thanks

- Antoine Spicher
- Olivier Michel

http://mgs.spatial-computing.org







- **PhD and other students**

  **Louis Bigo**

  J. Cohen, P. Barbier de Reuille,

  E. Delsinne, V. Larue, F. Letierce, B. Calvez,
  F. Thonerieux, D. Boussié *and the others...*

- **Past and presents Collaborations**
  - A. Lesne (IHES, stochastic simulation)
  - P. Prusinkiewicz (UoC, declarative modeling)
  - P. Barbier de Reuille (meristeme model)
  - C. Godin (CIRAD, biological modeling)
  - H. Berry (INRIA, stochastic simulation)
  - G. Malcolm (Liverpool, rewriting)
  - J.-P. Banâtre (IRISA, programming)
  - P. Fradet (Inria Alpes, programming)
  - F. Delaplace (IBISC, synthetic biology)
  - P. Dittrich (Jena, chemical organization)
  - F. Gruau (LRI, language and hardware)
  - P. Liehnard (Poitier, CAD, Gmap and quasi-manifold)