The MGS project

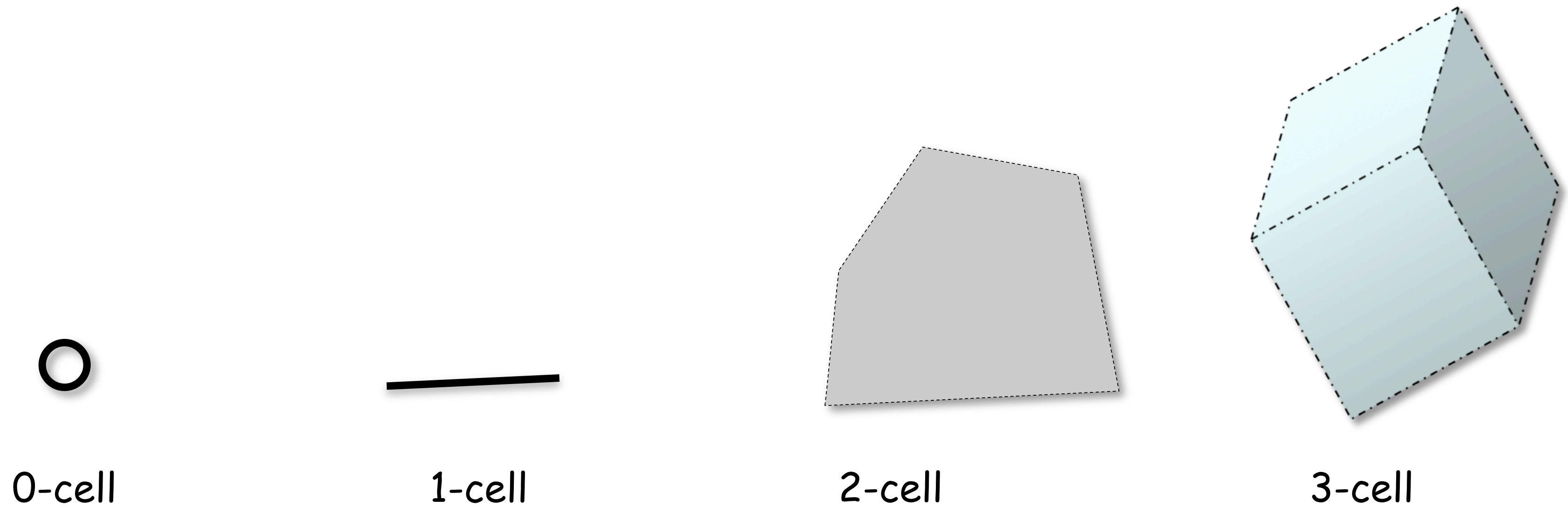
- Language dedicated to the simulation of $(DS)^2$
- Declarative (declarative simulation *vs* procedural)
- Abstract rewriting of complex spatial structures:
  - Data structure = topological collections

    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, …

Topological collection: representing the underlying space

Representation of space and structure

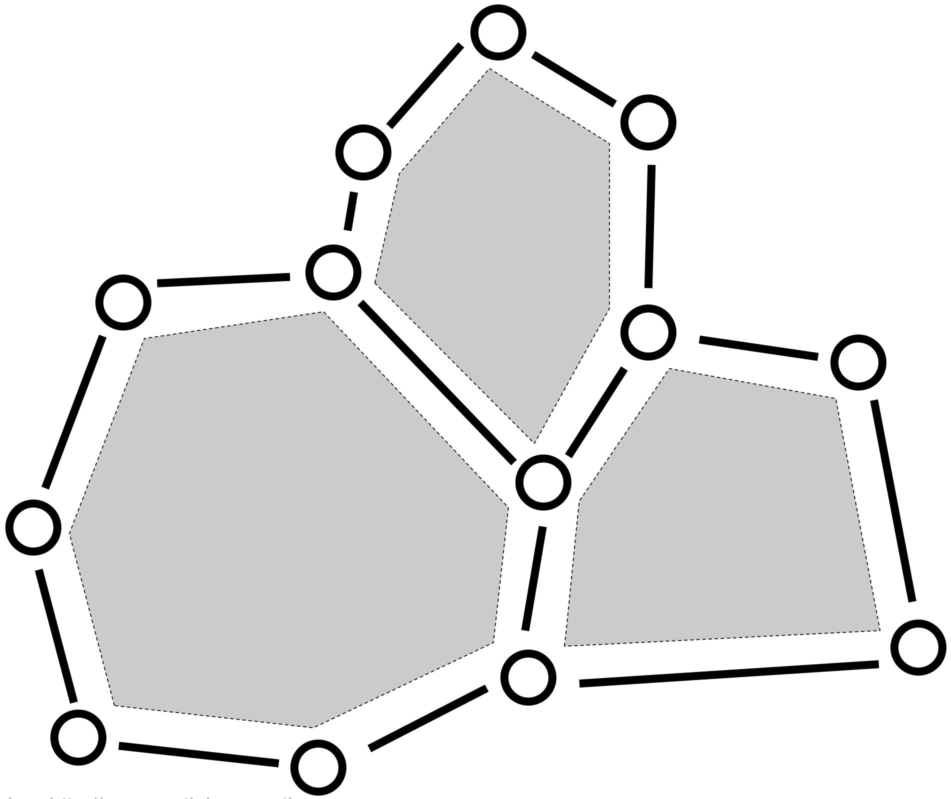
- Structure:
  - Collection of *topological cells*



Topological collection: representing the underlying space

Representation of space and structure

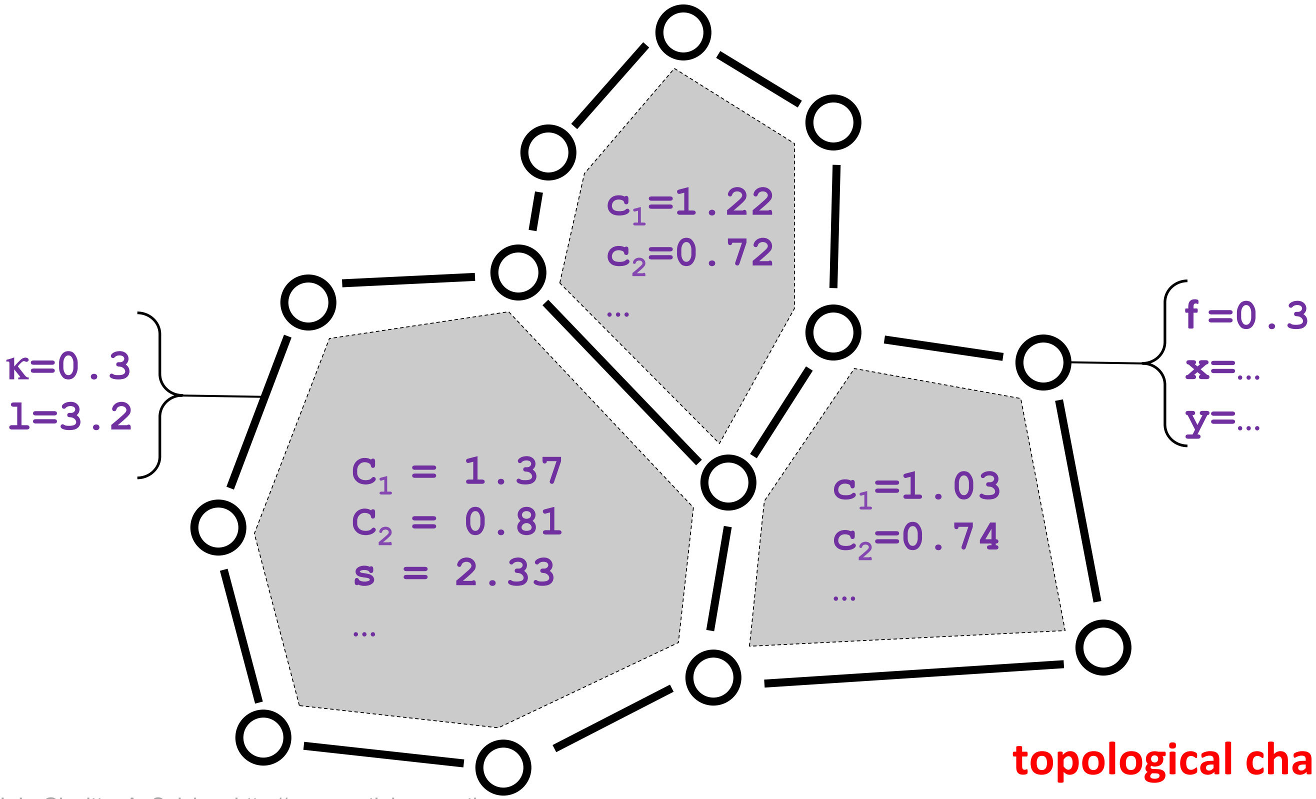
– Structure:
  - Collection of topological cells
  - *Incidence relationships*



Topological collection: a data-field over topological cells

Representation of space and structure

- Structure:
  - Collection of topological cells
  - Incidence relationship
- Data : *associating values* with topological cells ≈ <u>field in physics</u>

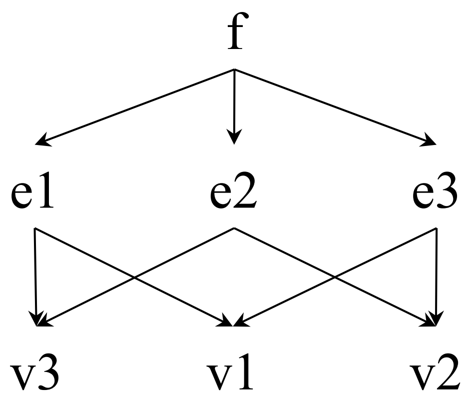


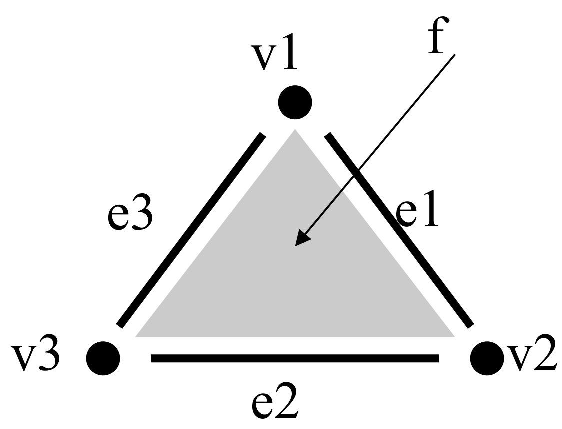
topological chain/cochain

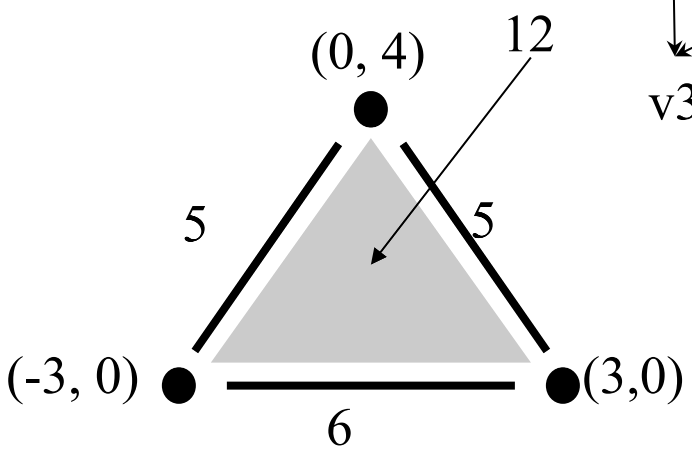
Abstract Simplicial Complex and simplicial chains

Incidence relationship and lattice of incidence:

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





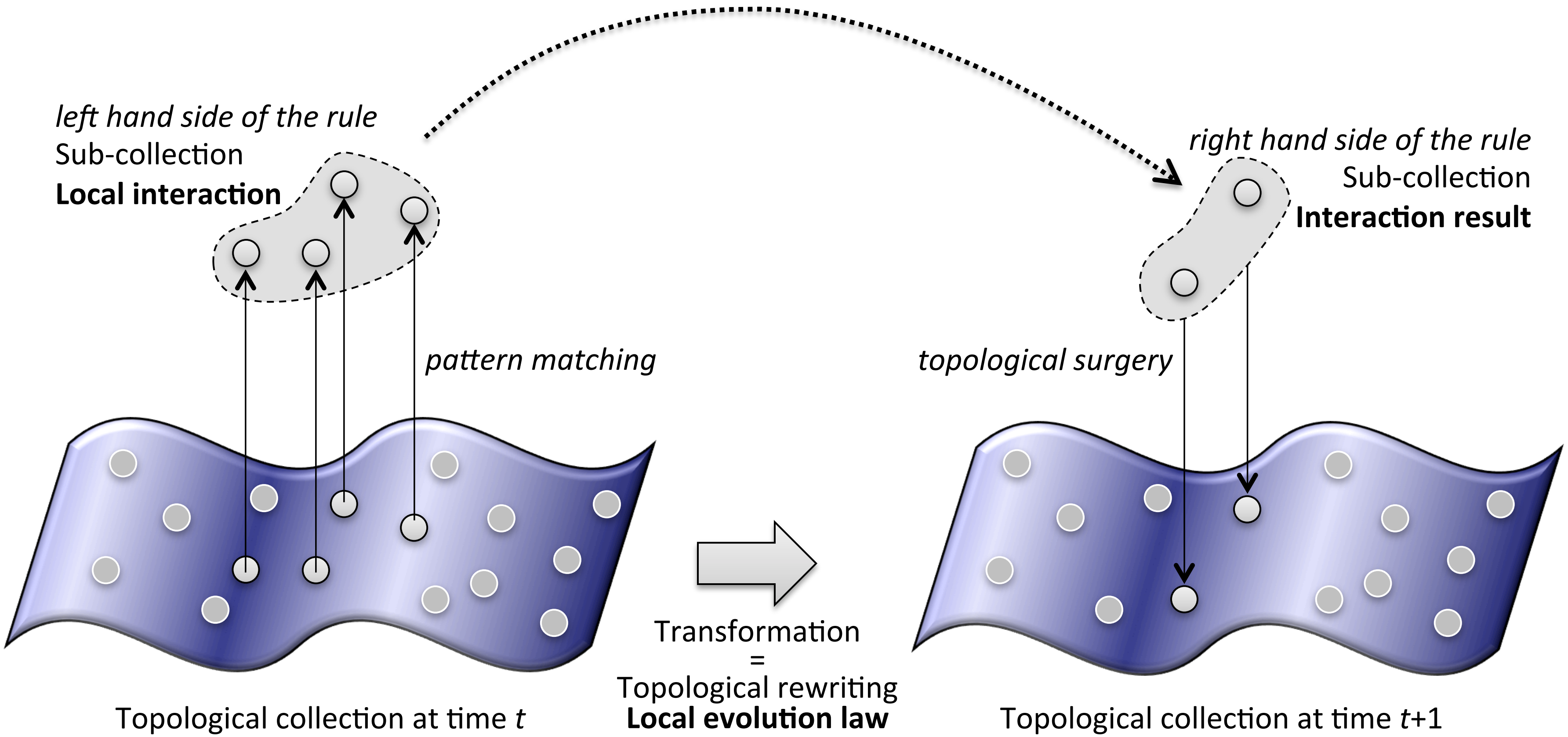


Topological chain

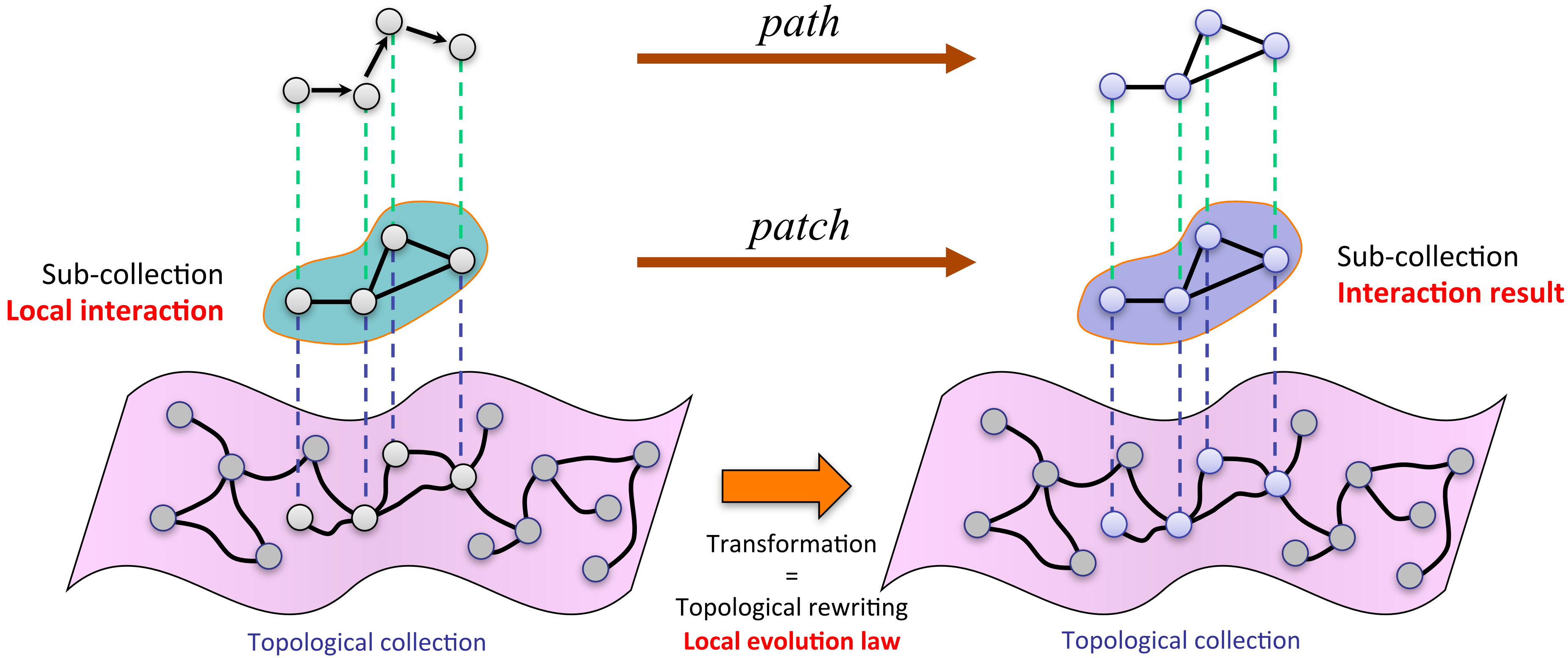
- coordinates with vertices
- lengths with edges
- area with f

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

Transformation



Transformation



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

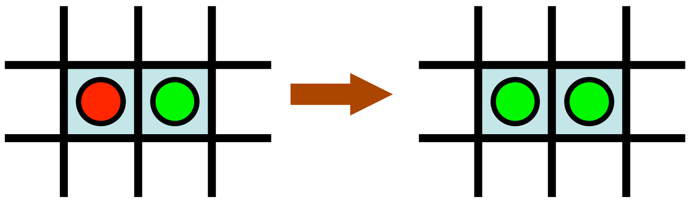
- *Path transformation* (path = sequence of neighbor elements)
  - Concise but limited expressiveness
- *Patch transformation* (arbitrary shape)
  - Longer but higher expressiveness

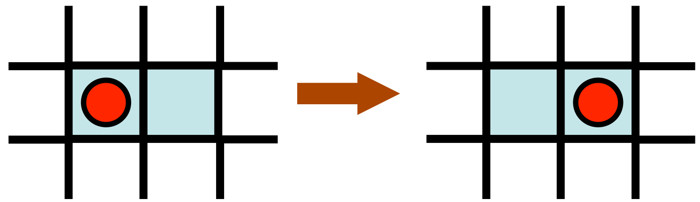
Example: Diffusion Limited Aggregation (DLA)

- 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



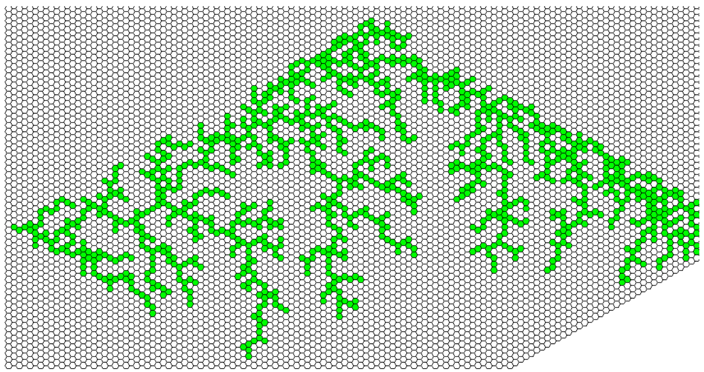


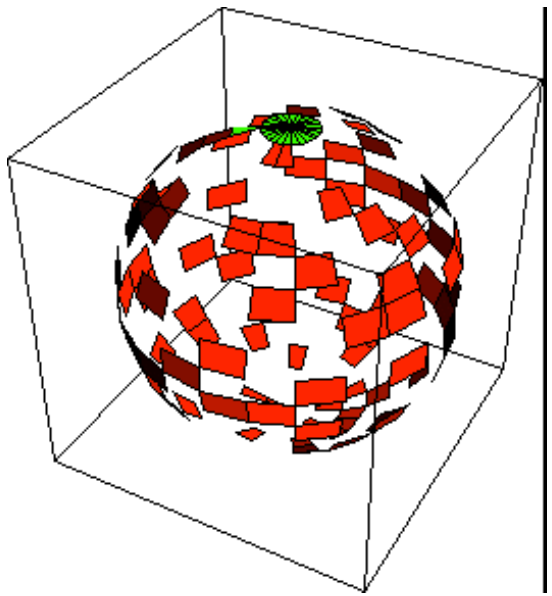
Example: Diffusion Limited Aggregation (DLA)

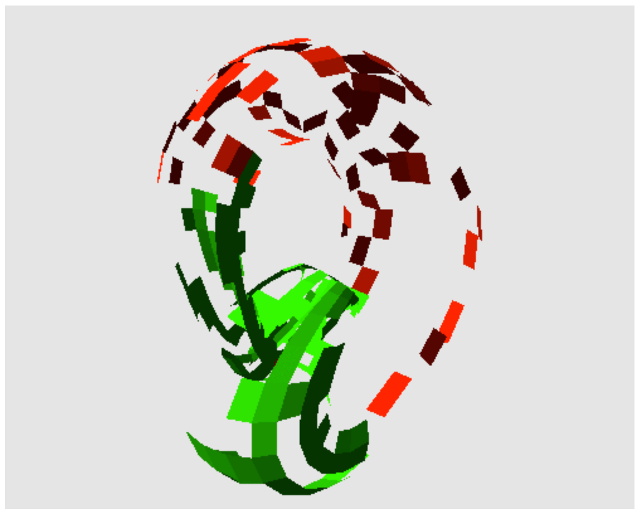
- Diffusion: some particles are randomly diffusing; others are fixed
- Aggregation: if a mobile particle meets a fixed one, it stays fixed

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

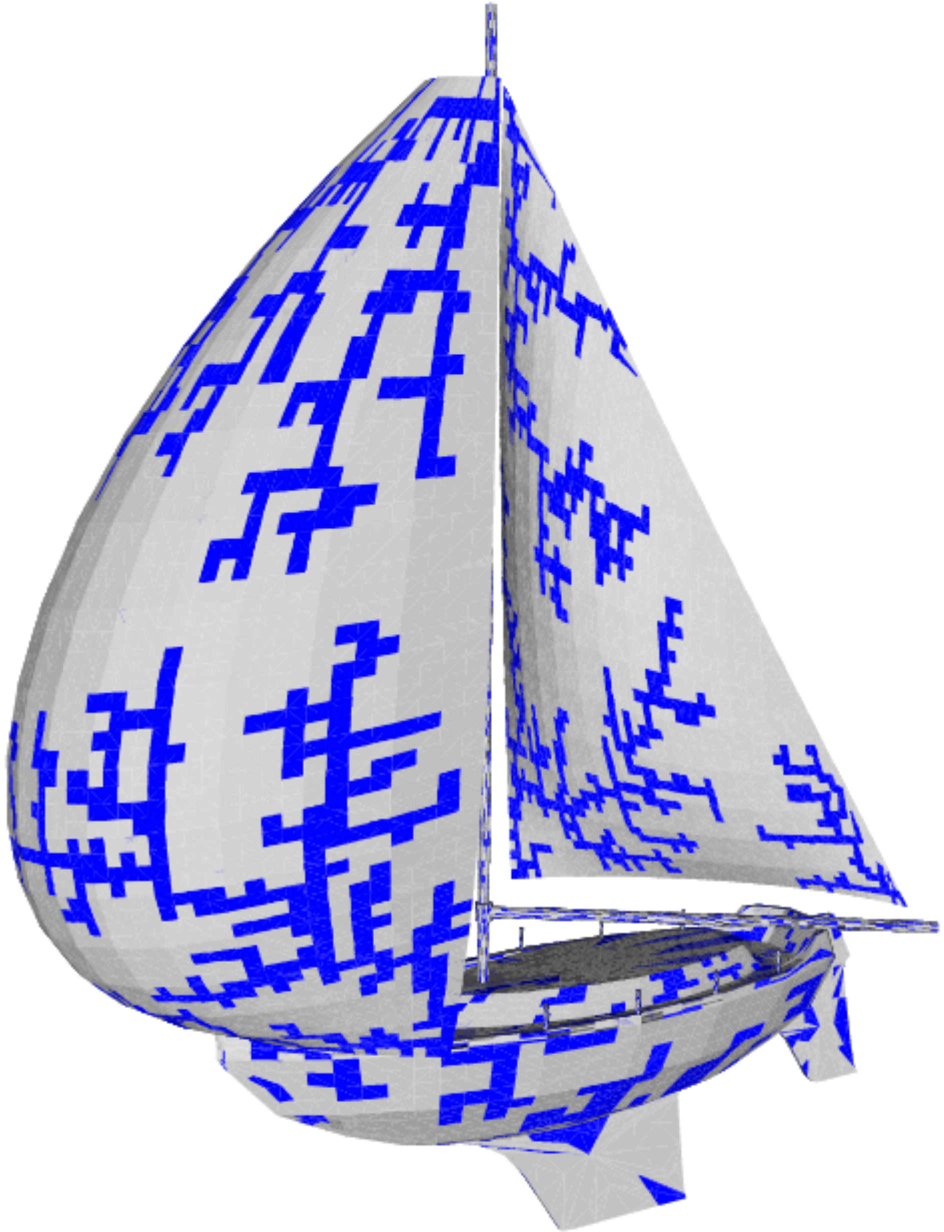
this transformation is an abstract process that can be applied to any kind of space







Polytypisme



Transformation = rewriting labeled cell complexes

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

arithmetic operation (the "+")

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

string concatenation (the ".")

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

multiset concatenation (= the chemical soup) (the "+")

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

gluing cell in a cell complex (the "+")

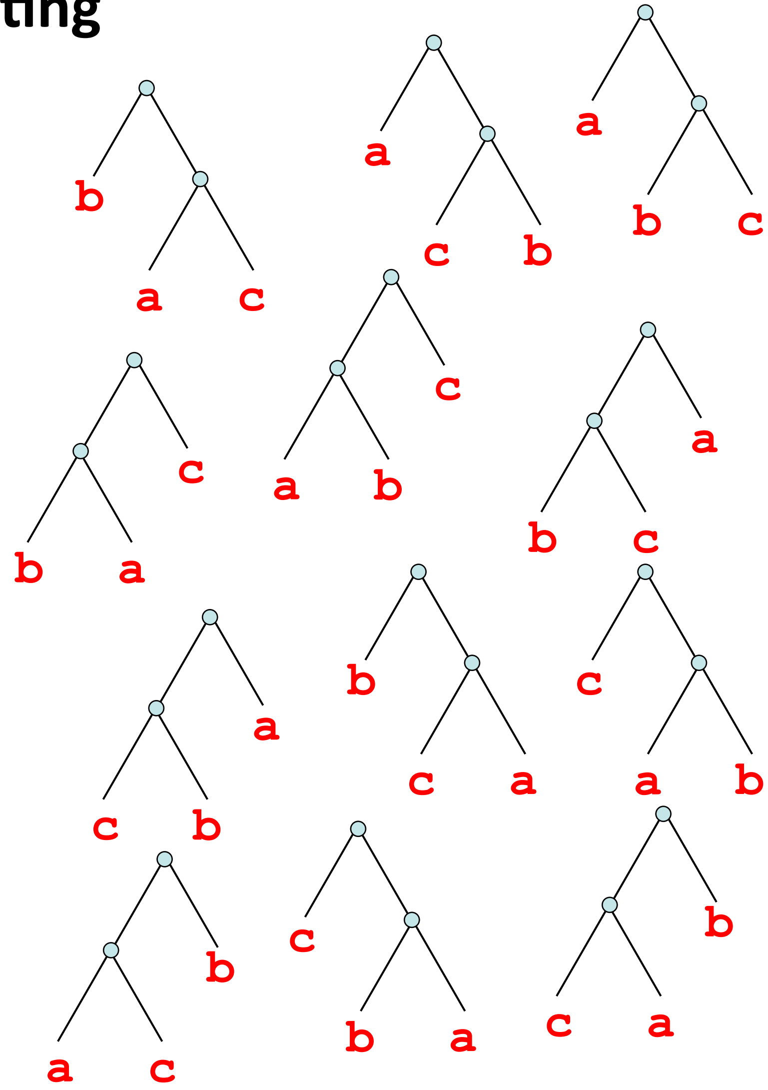
Trees and spatial structure

AC = stirring

- **Associative-commutative term rewriting**
  = multiset
  = *chemical soup*
  = chemical computing, P systems

- **Associative term rewriting**
  = string
  = *linear structure*
  = DNA computing, splicing systems

- **Term rewriting**
  = tree
  = *branching 1D structure*
  = L systems

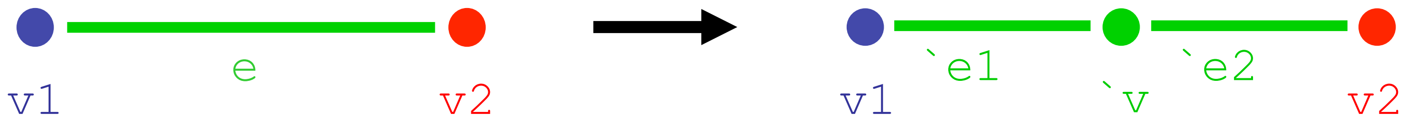


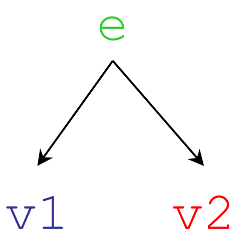
a(bc) = a(cb) = b(ac) = b(ca) = c(ab) = c(ba) = (ab)c = (ac)b = (ba)c = (bc)a = (ca)b = (cb)a
a(bc) = (ab)c
(a(bc) ≠ (ab)c

Topological rewriting ≠ graph rewriting

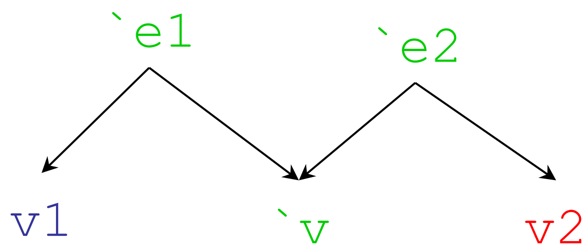
$v_1.\sigma_1 + v_2.\sigma_2 \to \ldots$ **topological rewriting (MGS)**

the structure is in the cells σ not in $+$



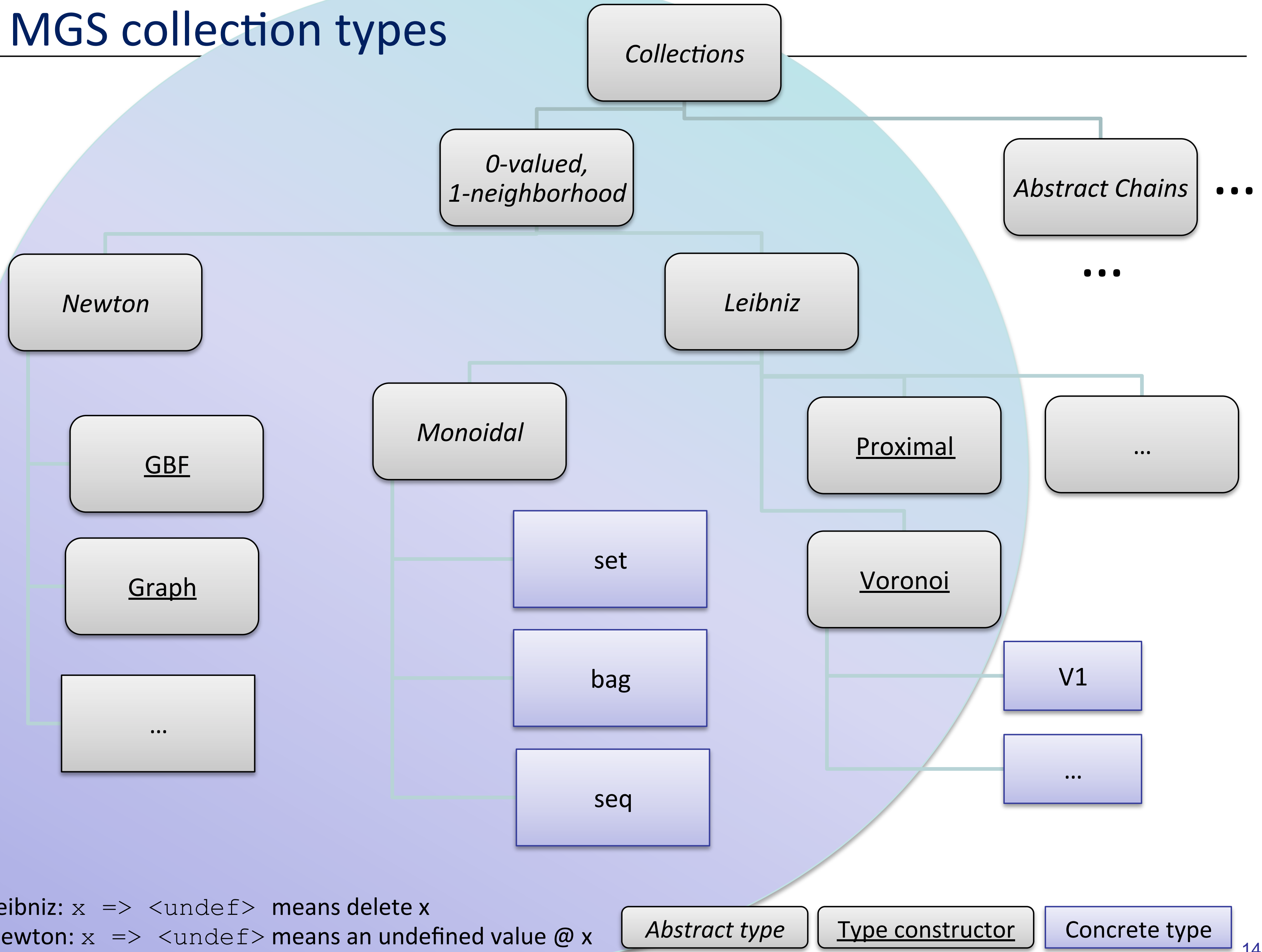


$x_1.v_1 + (x.e + x_2.v_2)$
$= (x_1.v_1 + x.e) + x_2.v_2$
$= (x_1.v_1 + x_2.v_2) + x.e$
$= x.e + (x_1.v_1 + x_2.v_2)$
$= \ldots$




```
v1 < e:[ dim = 1 ] > v2 =>
   v1
   `e1:[ dim = 1,    faces = (^v1,`v),  val = … ]
   `v :[ dim = 0, cofaces = (`e1,`e2), val = (v1+v2)/2 ]
   `e2:[ dim = 1,    faces = (^v2,`v),  val = … ]
   v2
```

MGS collection types



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

0-valued, 1-neighborhood Collection

Graph = 1D Simplicial Complex

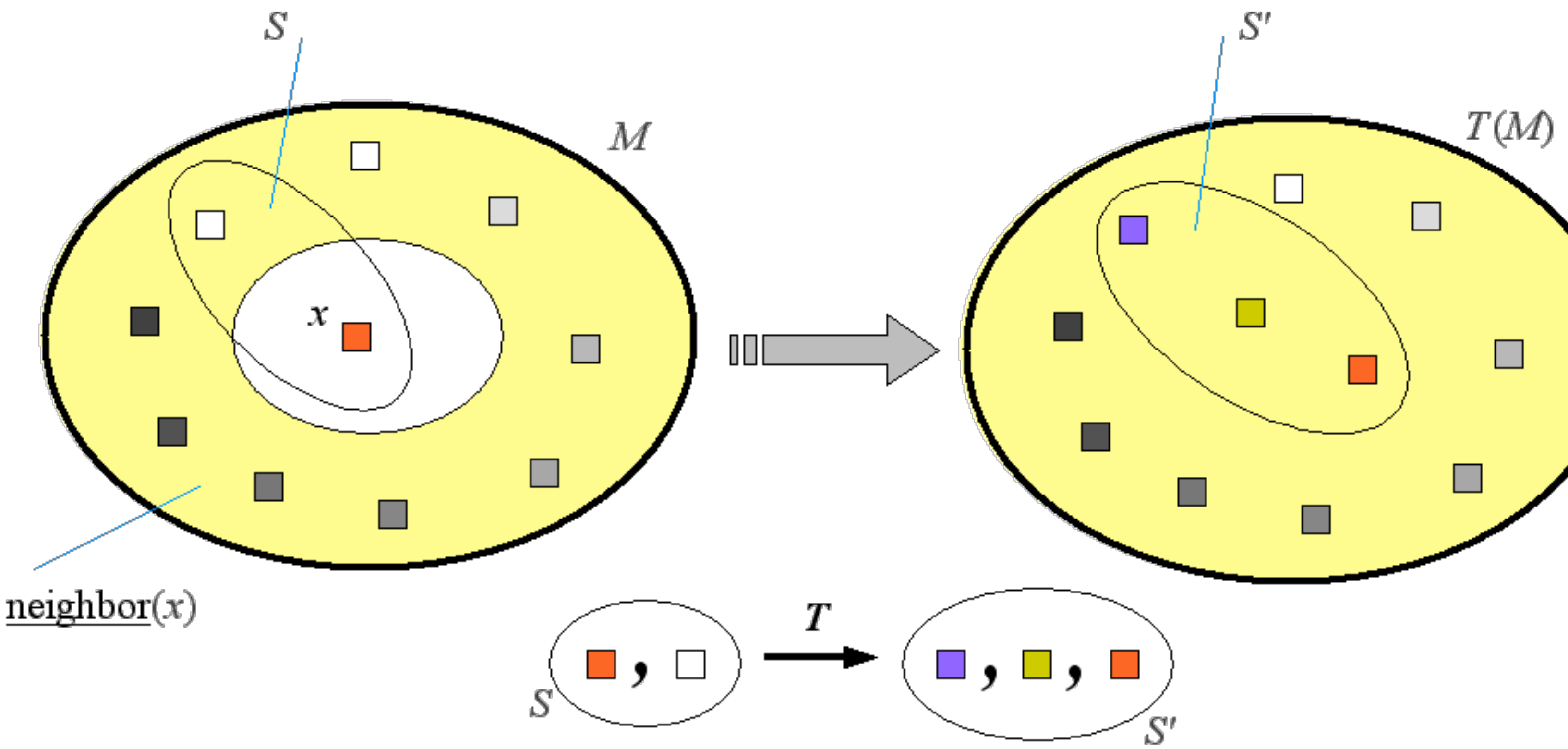
- Vertices are labeled
- The neighborhood is defined by the edge of a graph

- Isolated graphs (no edges) : record
- Complete graphs : set and multisets
- Linear:
  - sequence (list)
  - ring
- Uniform graphs: Cayley graphs

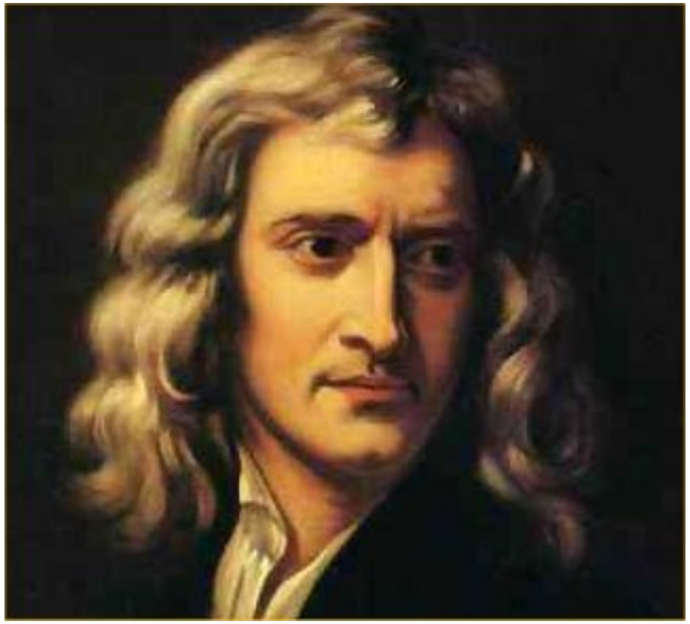
  the graphical representation of a group presentation
- Graph defined by a distance:
  - proximal
  - delaunay

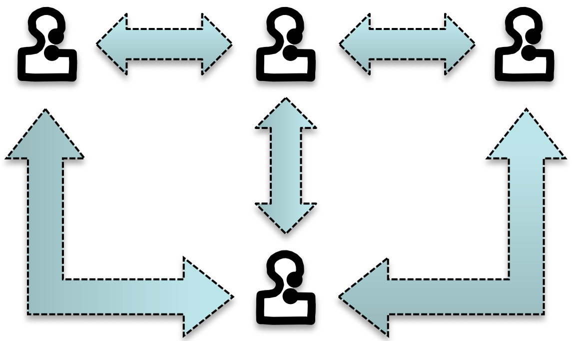
Complete graphs, multiset and concrete topology

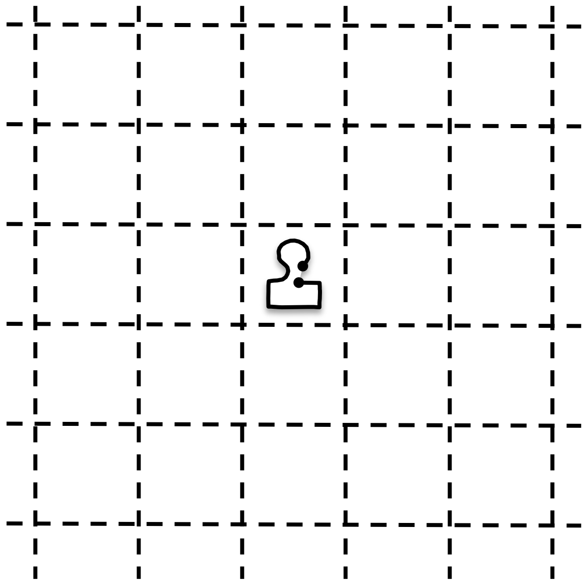
collection = multiset M
topology : neighbor(x) = M - $\{x\}$ *any element is neighbor of any other element*
subcollection S = prefixe of an orbite of neighbor = multiset
boundary(S) = S.



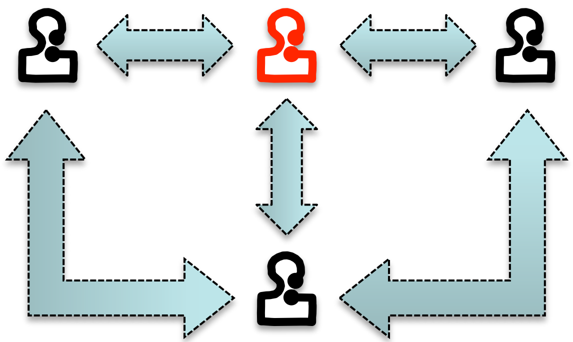
Leibniz *vs.* Newton



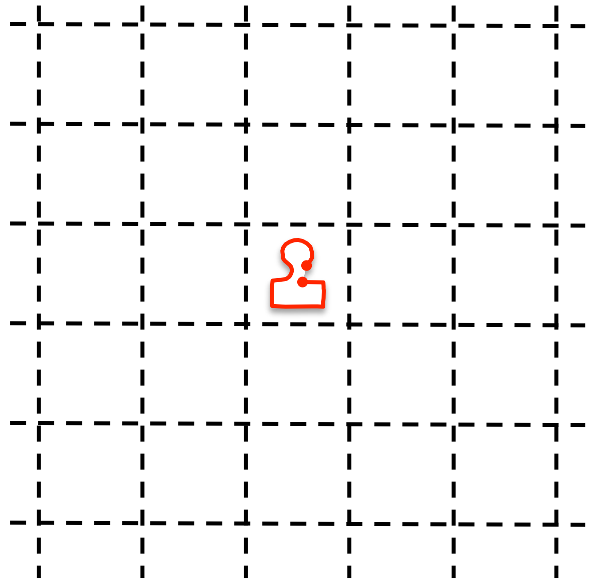


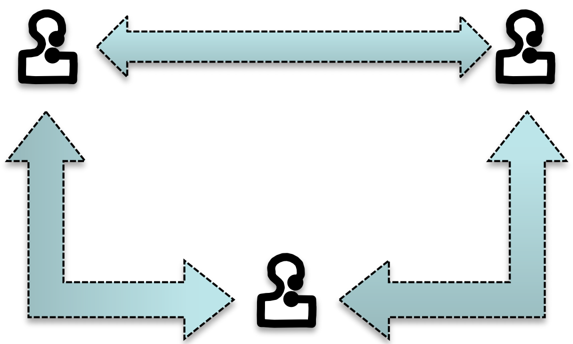


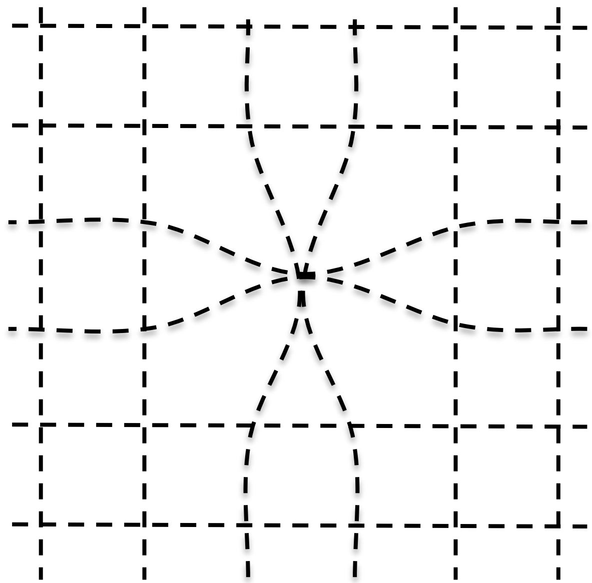
Leibniz *vs.* Newton



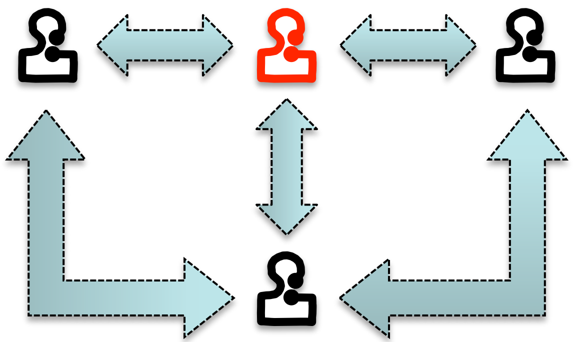
x => .

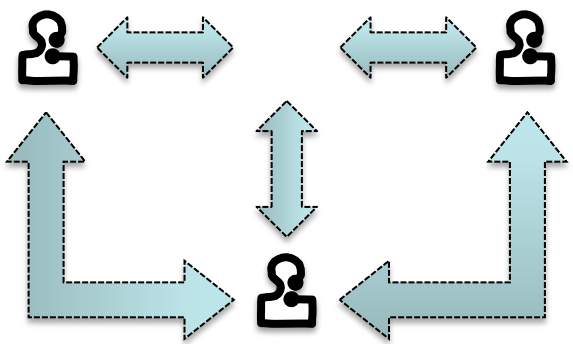


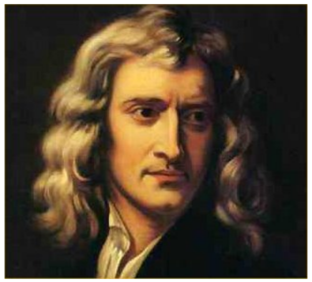




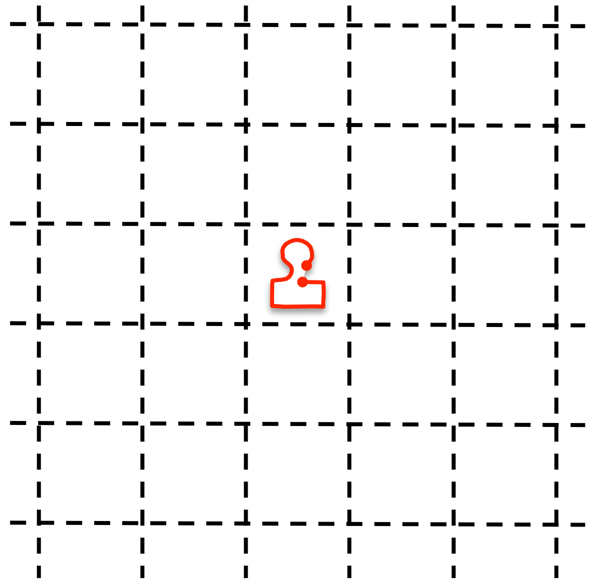
Leibniz *vs.* Newton

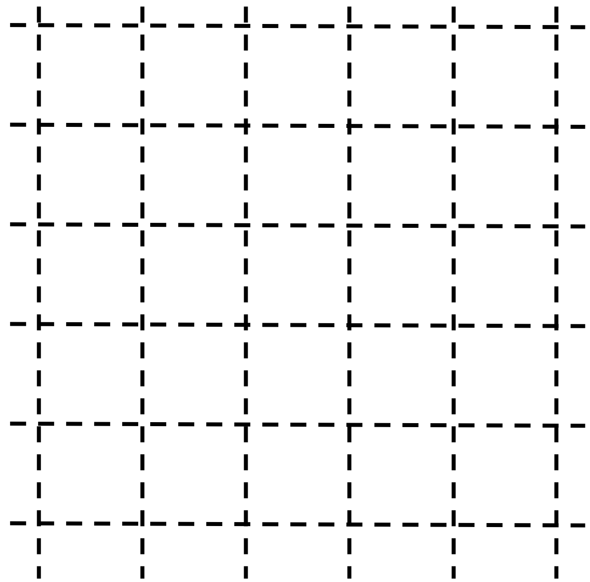






x => .





Acknowledgements

MGS: Antoine Spicher, Olivier Michel, Julien Cohen

S&S Bio : Hanna Klaudel, Franck Delaplace, Hugues Berry, Przemek Prusinkiewicz, Annick Lesne…

Spatial Computing: Jacob Beal, Fréderic Gruau, René Doursat…

Examples: Pierre Barbier de Reuille, Christophe Godin, Samuel Bottani, the Paris iGEM'07 team…

Some figures are borowed from Olivier Michel, Antoine Spicher, Pierre Barbier de Reuille, Franck Delaplace, Hugues Berry (INRIA), the iGEM Paris 2007 and many others.