

---

# Topological Rewriting & Patch Transformations

Antoine Spicher

[www.spatial-computing.org/mgs](http://www.spatial-computing.org/mgs)

SUPMECA

June 2015

# Outline



- MGS: a Formal Introduction
- Patch Transformations
- Differential Operators
- An Integrative Example: T-shape growth

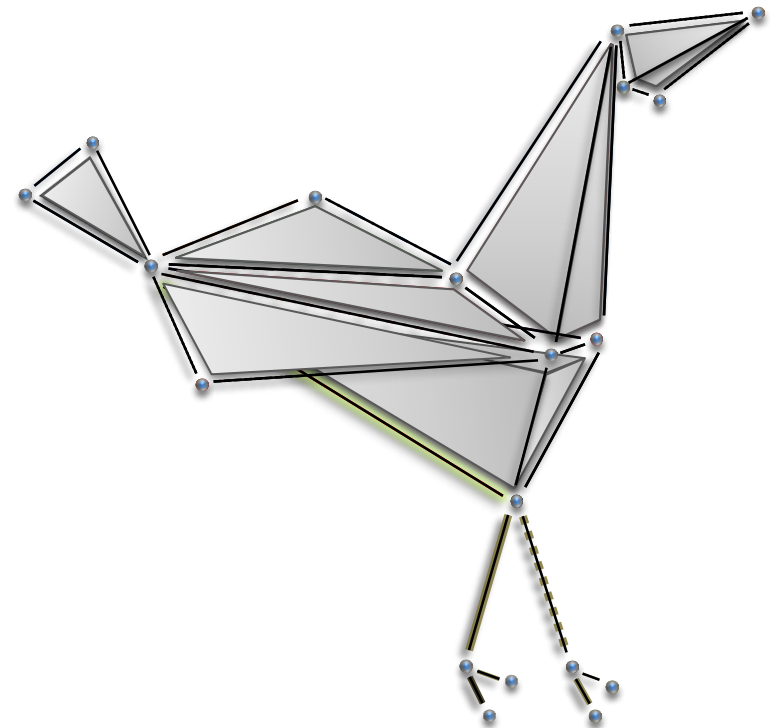
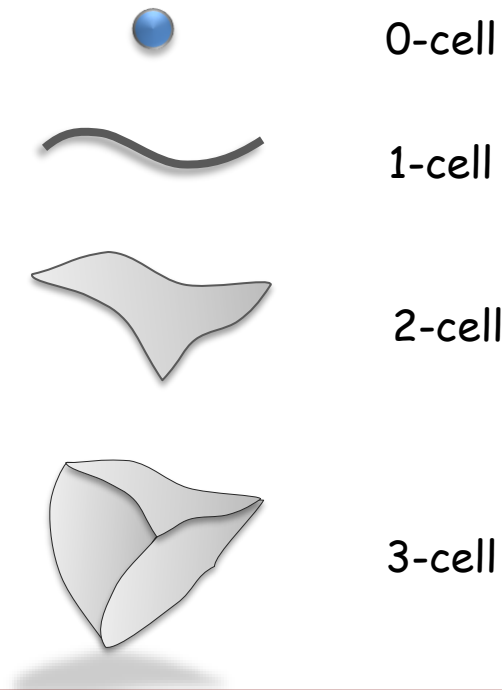
# MGS Formalism: Collection

## ■ Topological Collection

### □ Structure

- A collection of *(topological) cells*
- An *neighborhood relationship*

### □ Data *associated with the cells*



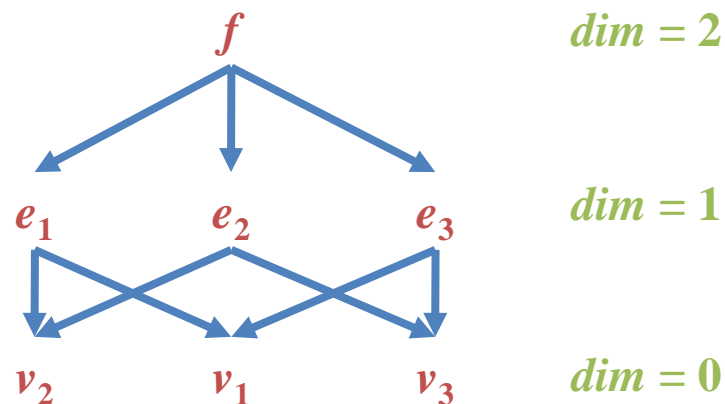
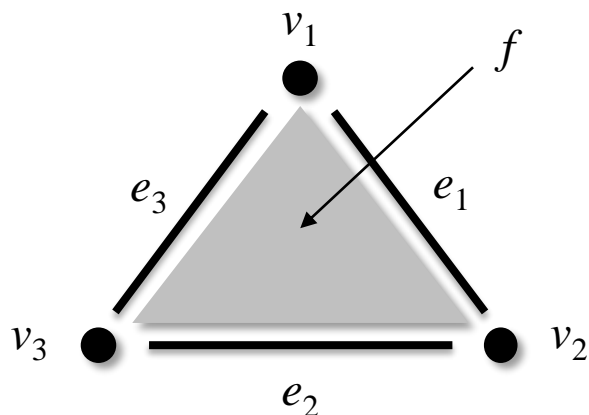
# MGS Formalism: Collection

## ■ Abstract Cellular Complex (ACC)

Let  $(S_n)$  be a family of disjoint sets of symbols called **topological cells**.  
The **dimension**  $n$  of a cell  $\sigma \in S_n$  is denoted  $\dim(\sigma)$ . We write  $S = \bigcup_n (S_n)$ .

An **abstract cellular complex**  $\mathcal{K}$  on  $S$  is a couple  $(S, <)$  such that

- $S \subset S$  is a **set of topological cells**,
- $< \subset S \times S$  is a partial order on  $S$ , called the **incidence relation**,
- the **dimension** is **monotonic** for  $<$ :  $\sigma_1 < \sigma_2 \Rightarrow \dim(\sigma_1) < \dim(\sigma_2)$ .



# MGS Formalism: Collection

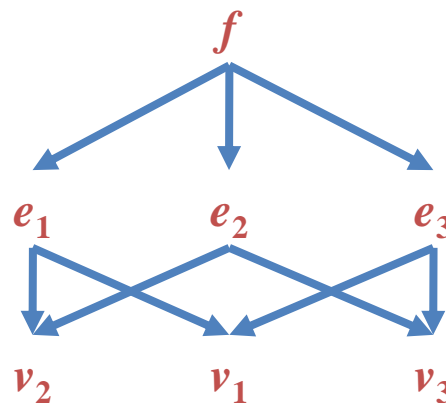
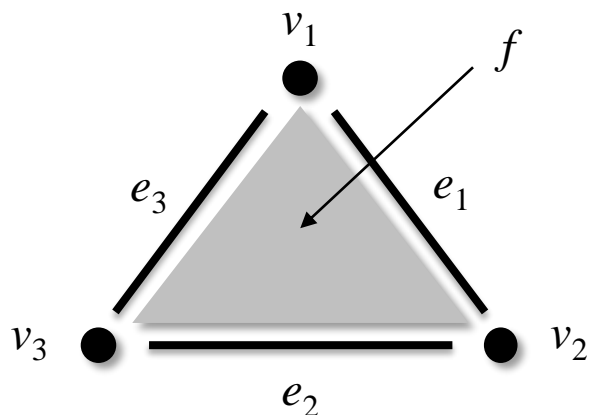
## ■ Some neighborhoods on ACC

### □ Face/Coface relationship ( $<$ , $>$ )

Let  $\mathcal{K}$  be an ACC and let  $\sigma$  and  $\tau$  be two cells of  $\mathcal{K}$ .

The cell  $\tau$  is called **face of**  $\sigma$  if  $\tau < \sigma$  and  $\dim(\tau) = \dim(\sigma) - 1$ .

The cell  $\sigma$  is called **coface of**  $\tau$ . This relation is denoted by  $\tau < \sigma$ .



$$\begin{aligned} v_1 &< e_1 > v_2 \\ v_2 &< e_2 > v_3 \\ v_3 &< e_3 > v_1 \\ e_1 &> f \\ e_2 &> f \\ e_3 &> f \end{aligned}$$

# MGS Formalism: Collection

## ■ Some neighborhoods on ACC

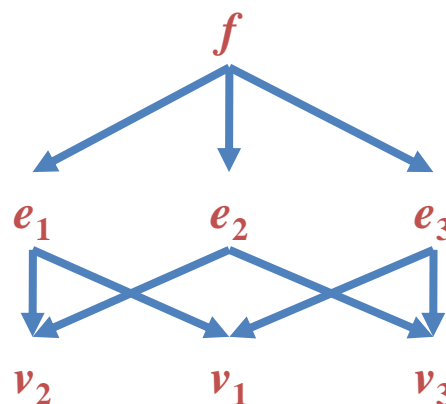
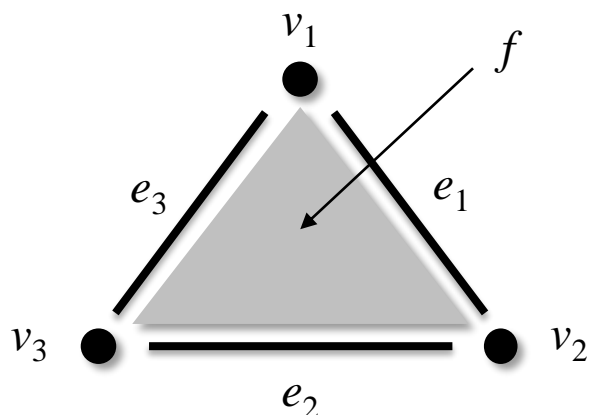
- Face/Coface relationship ( $<, >$ )
- $p$ -Neighborhood ( $,_p$ )

Let  $\mathcal{K}$  be an ACC,  $\tau_1$  and  $\tau_2$  two  $n$ -cells of  $\mathcal{K}$  and  $p$  an integer.

The cells  $\tau_1$  and  $\tau_2$  are said  **$p$ -neighbors** if there exists  $\sigma \in \mathcal{K}$  such that

- $\tau_1 > \sigma$  and  $\tau_2 > \sigma$  if  $n > p$ , or
- $\tau_1 < \sigma$  and  $\tau_2 < \sigma$  if  $n < p$

This relation is denoted by a comma:  $\tau_1 ,_p \tau_2$ .



$v_1 ,_1 v_2$   
 $v_2 ,_2 v_3$   
 $e_3 ,_0 e_1$   
 $e_2 ,_2 v_1$   
 $\dots$

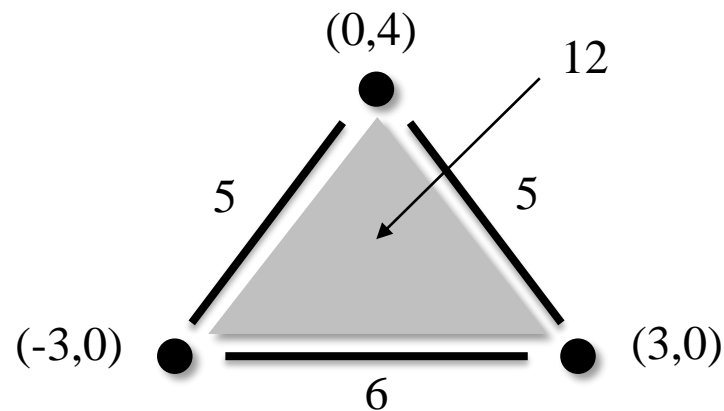
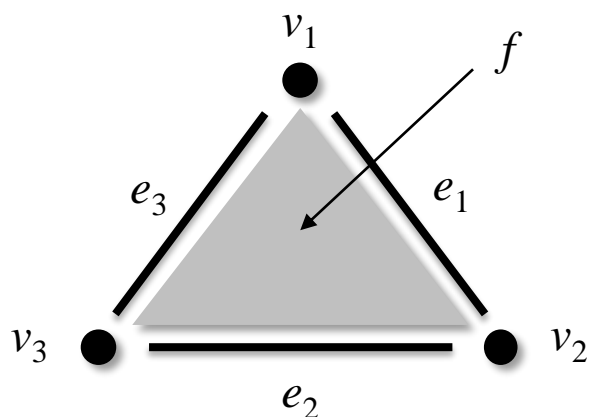
# MGS Formalism: Collection

## ■ Labeling of an ACC

Let  $\mathcal{K}$  be an ACC and let  $V$  be an arbitrary set of values.

A **topological collection** over  $\mathcal{K}$  with values in  $V$  is a partial function from  $\mathcal{K}$  to  $V$ .  $\mathcal{C}_{\mathcal{S}}(V)$  denotes the set of collections with values in  $V$ .

$$c = (0,4).v_1 + (3,0).v_2 + (-3,0).v_3 + 5.e_1 + 6.e_2 + 5.e_3 + 12.f$$

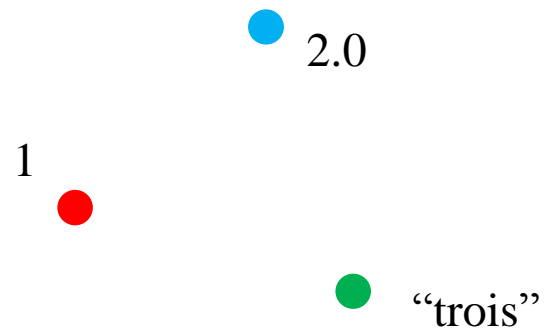


# MGS Formalism: Collection

## ■ Types of collections

- Depending on the topology of the underlying cellular complex
- Records (equivalent to a C struct)
  - Let  $\mathcal{F}$  be the set of fields
  - $\mathcal{K} = (F, \emptyset)$  with  $F \subset \mathcal{F}$ , a totally disconnected space

{ **a** = 1, **b** = 2.0, **c** = "trois" }

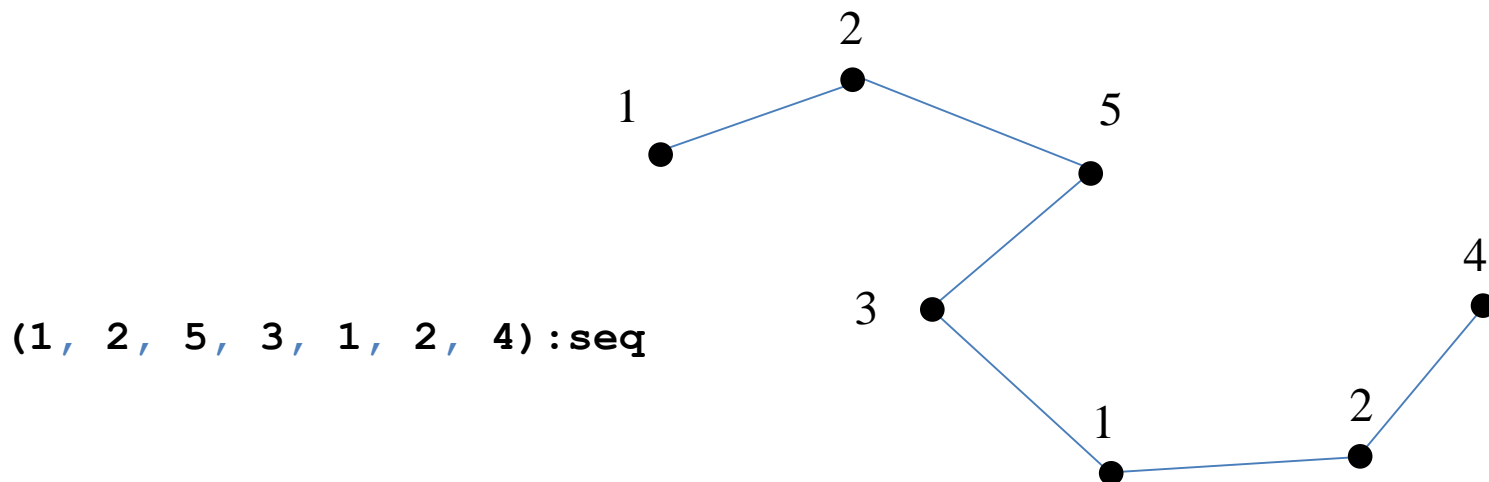




# MGS Formalism: Collection

## ■ Types of collections

- Depending on the topology of the underlying cellular complex
- Monoidal collections
  - Collections builds from singleton and **join** operator
  - Topology depends on the properties of the **join** operator
    - Sequence (**associative**): **linear graph**

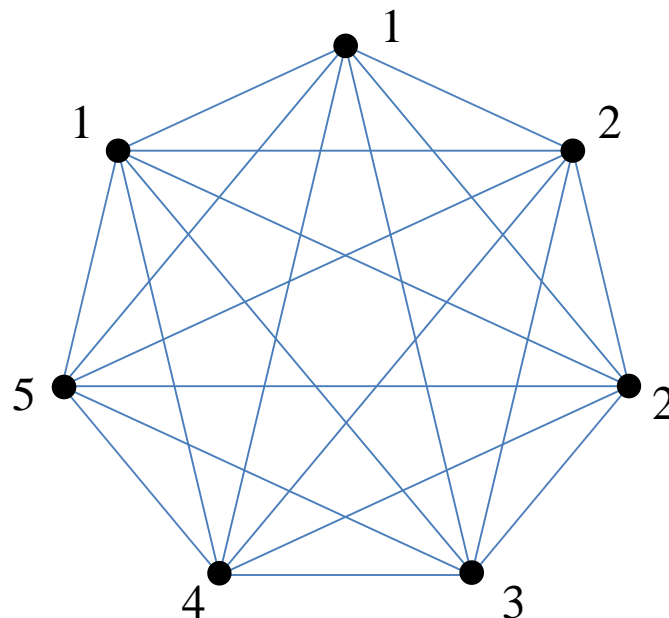


# MGS Formalism: Collection

## ■ Types of collections

- Depending on the topology of the underlying cellular complex
- Monoidal collections
  - Collections builds from singleton and **join** operator
  - Topology depends on the properties of the **join** operator
    - Sequence (**associative**): **linear graph**
    - Bag (**associative/commutative**): **complete graph**

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

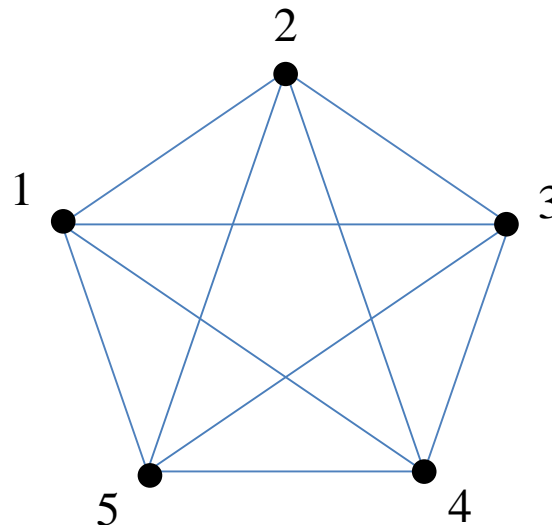


# MGS Formalism: Collection

## ■ Types of collections

- Depending on the topology of the underlying cellular complex
- Monoidal collections
  - Collections builds from singleton and **join** operator
  - Topology depends on the properties of the **join** operator
    - Sequence (**associative**): **linear graph**
    - Bag (**associative/commutative**): **complete graph**
    - Set (**associative/commutative/idempotent**): **complete graph**

(1, 2, 5, 3, 1, 2, 4) : **set**



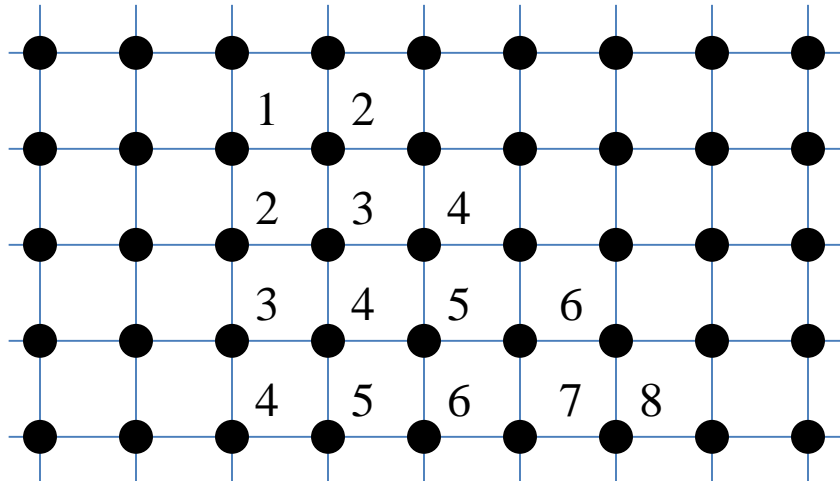
# MGS Formalism: Collection

## ■ Types of collections

- Depending on the topology of the underlying cellular complex
- GBF collections
  - Let  $G = \langle d_1, d_2, \dots \mid r_1, r_2, \dots \rangle$  be a finitely generated group
  - $\mathcal{K} = (G, \{ (g, g \pm d_i) \mid g \in G \})$ , the **Cayley's graph** of  $G$

```
gbf NEWS = < north, east, west, south ;  
            north = -south, east = -west > ;;
```

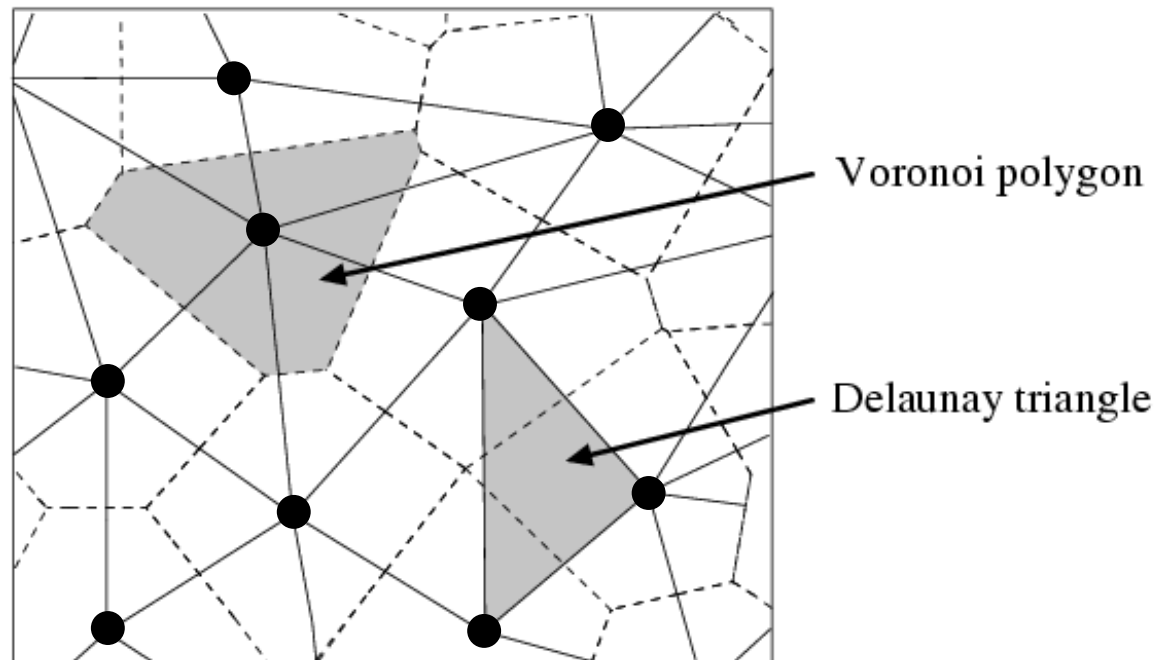
```
c = (
    ( 1, 2 ),
    ( 2, 3, 4 ),
    ( 3, 4, 5, 6 ),
    ( 4, 5, 6, 7, 8 )
) following
    |south>, |east> ;;
```



# MGS Formalism: Collection

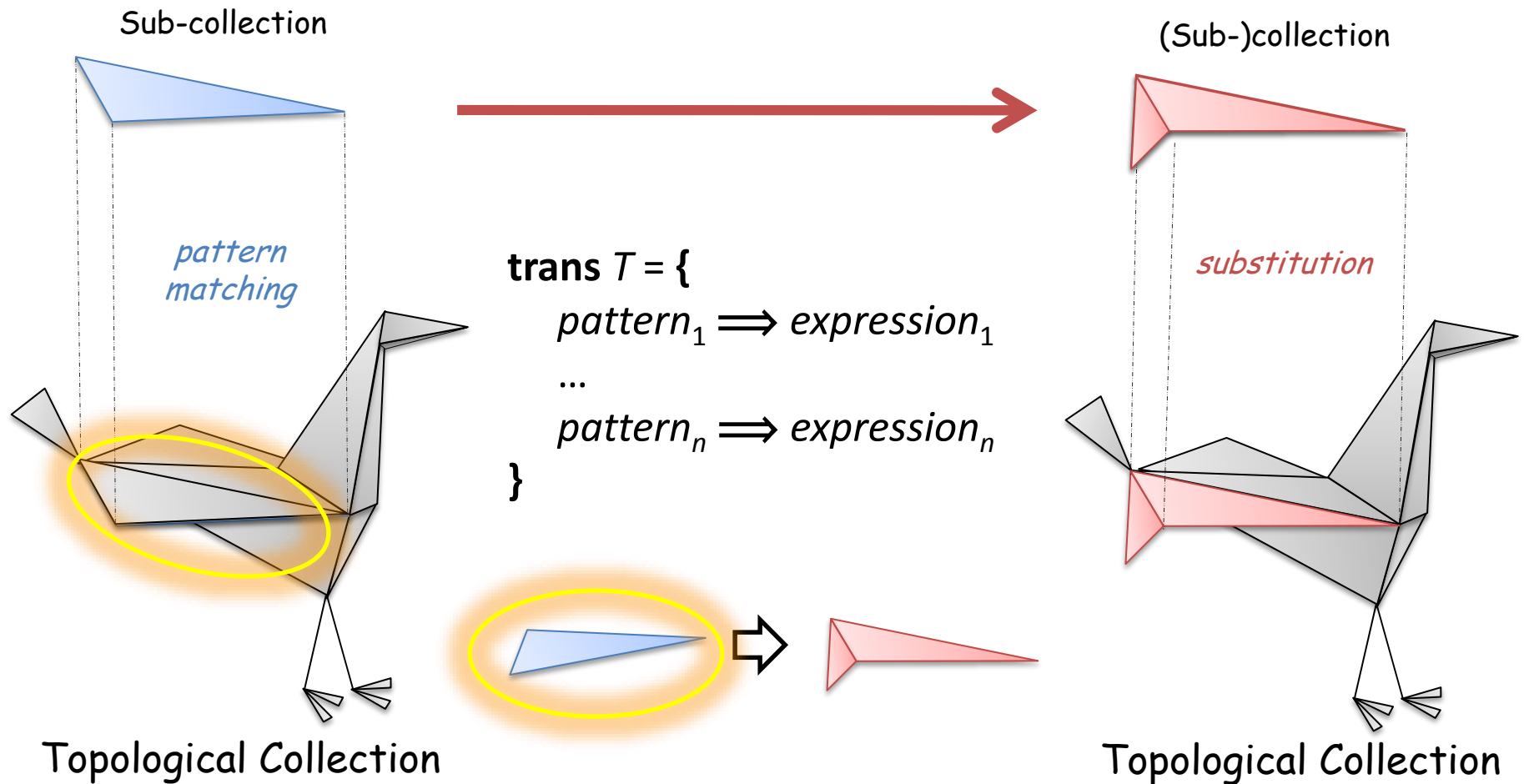
## ■ Types of collections

- Depending on the topology of the underlying cellular complex
- Delaunay collections
  - Built from a Voronoi tessellation of a set of points
  - Association of a region of space with each node



# MGS Formalism: Transformation

## ■ Transformation



# MGS Formalism: Transformation

## ■ Transformation

- Function of collections defined by case
- Each case is specified by a *rule*

*pattern*  $\Rightarrow$  *expression*

- Semantics of a transformation: *topological rewriting*

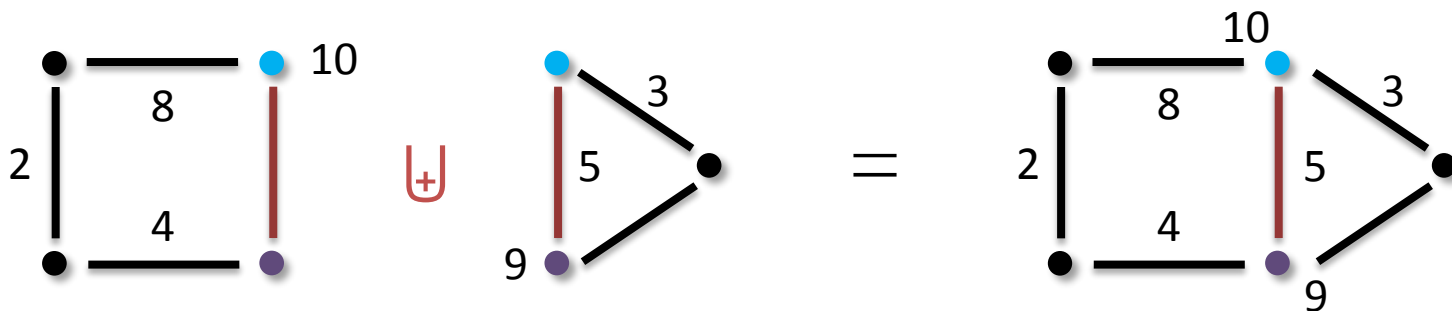
## ■ Requirements

- Topological collection patterns
- Topological collection expressions, environments and evaluation
- Pattern matching
- Rewriting rule/relation

# MGS Formalism: Transformation

## ■ Some notations

- Collection:  $c = \sum v. \sigma$
- Shape:  $\text{Shape}(c) = \mathcal{K}$
- Support:  $|c| = \{ \sigma \in \text{Shape}(c) \mid c(\sigma) \text{ is defined} \}$
- Extension:  $c' = c|_{\mathcal{K}}$   
 $c'(\sigma) = c(\sigma)$  when  $\sigma \in \mathcal{K} \cap \text{Shape}(c)$ , and is undefined on  $\mathcal{K} - \text{Shape}(c)$
- Merge:  $c_1 \sqcup c_2$   
 $c_1|_{\mathcal{K}} + c_2|_{\mathcal{K}}$  where  $\mathcal{K} = \text{Shape}(c_1) \cup \text{Shape}(c_2)$





# MGS Formalism: Transformation

## ■ Topological Collection Patterns

□ Let consider the two sets of variables:

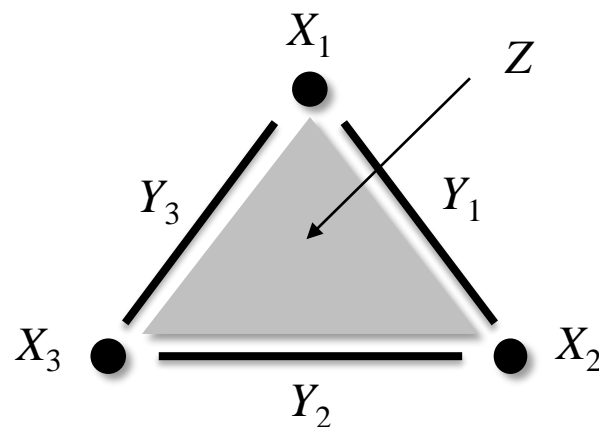
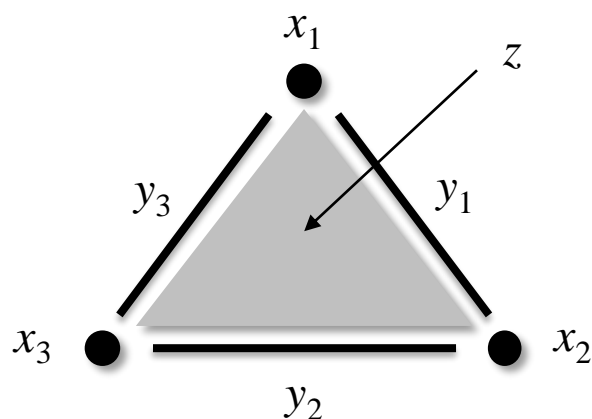
■  $S^{\text{var}} = \{x_1, x_2, \dots\}$  variables denoting cells

Elements of are ranked by dimension (i.e.,  $S^{\text{var}} = \bigcup_n S_n^{\text{var}}$ )

■  $V^{\text{var}} = \{X_1, X_2, \dots\}$  variables denoting values

□ A **pattern** is a topological collection of  $C_{S^{\text{var}}}(V^{\text{var}})$

$$\alpha = X_1.x_1 + X_2.x_2 + X_3.x_3 + Y_1.y_1 + Y_2.y_2 + Y_3.y_3 + Z.z$$



# MGS Formalism: Transformation

## ■ Topological Collection Expressions

- Similar to topological collection patterns
  - Extending value variables with expressions  $\Sigma$
  - A *collection expression* is a collection of  $C_{S^{\text{var}} \cup S}(\Sigma)$
- Environments
  - $\Gamma_S = S^{\text{var}} \rightarrow S$
  - $\Gamma_V = V^{\text{var}} \rightarrow V$
- Evaluation function

$$\zeta: C_{S^{\text{var}} \cup S}(\Sigma) \times \Gamma_S \times \Gamma_V \rightarrow C_S(V)$$

# MGS Formalism: Transformation

## ■ Pattern Matching

- A pattern  $\alpha = X_1.x_1 + \dots + X_m.x_m$   
*pattern-matches* a collection  $c$   
*with environments*  $\rho_S \in \Gamma_S$  and  $\rho_V \in \Gamma_V$  iff

$$c = \rho_V(X_1). \rho_S(x_1) + \dots + \rho_V(X_m). \rho_S(x_m)$$

- A pattern  $\alpha$  *matches a collection*  $c'$  in a collection  $c$   
*with environments*  $\rho_S \in \Gamma_S$  and  $\rho_V \in \Gamma_V$  iff
  - $c'|_{\text{Shape}(c)}$  is a sub-collection of  $c$
  - $\text{Shape}(c') \subset \text{Shape}(c)$
  - $\alpha$  *pattern-matches*  $c'$  *with environments*  $\rho_S$  and  $\rho_V$

# MGS Formalism: Transformation

## ■ Rewriting rule & rewriting relation

- Rewriting rule  $\alpha \Rightarrow \beta$ 
  - $\alpha$  is a topological collection pattern
  - $\beta$  is a topological collection expression
- One-step rewriting relation:  $c_1 \triangleright_{\alpha \Rightarrow \beta} c_2$  iff
  - $c_1 = l \uplus c$  ( $l$  is the redex and  $c$  is the context)  
such that  $\alpha$  matches  $l$  in  $c_1$  with some environments  $\rho_s$  and  $\rho_v$
  - $c_2 = r \uplus c$   
such that  $r = \zeta(\beta, \rho_s, \rho_v)$
  - $(\text{Shape}(r) - \text{Shape}(l)) - \text{Shape}(c) = \emptyset$
- $\triangleright_R$  trivial extension to a set  $R$  of rules

# MGS Formalism: Transformation

## ■ Topological Rewriting

□  $\triangleright_R$  *parallel rewriting* of a set  $R$  of rules

$$\begin{array}{ccccccc}
 c_1 & = & l_1 & \uplus & \dots & \uplus & l_n & \uplus & c \\
 \downarrow \triangleright_R & & \downarrow \triangleright_R & & & & \downarrow \triangleright_R & & \downarrow \triangleright_R \\
 c_2 & = & r_1 & \uplus & \dots & \uplus & r_n & \uplus & c
 \end{array}$$

$$|l_i| \cap |l_j| = \emptyset \text{ for all } i \neq j$$

# Outline

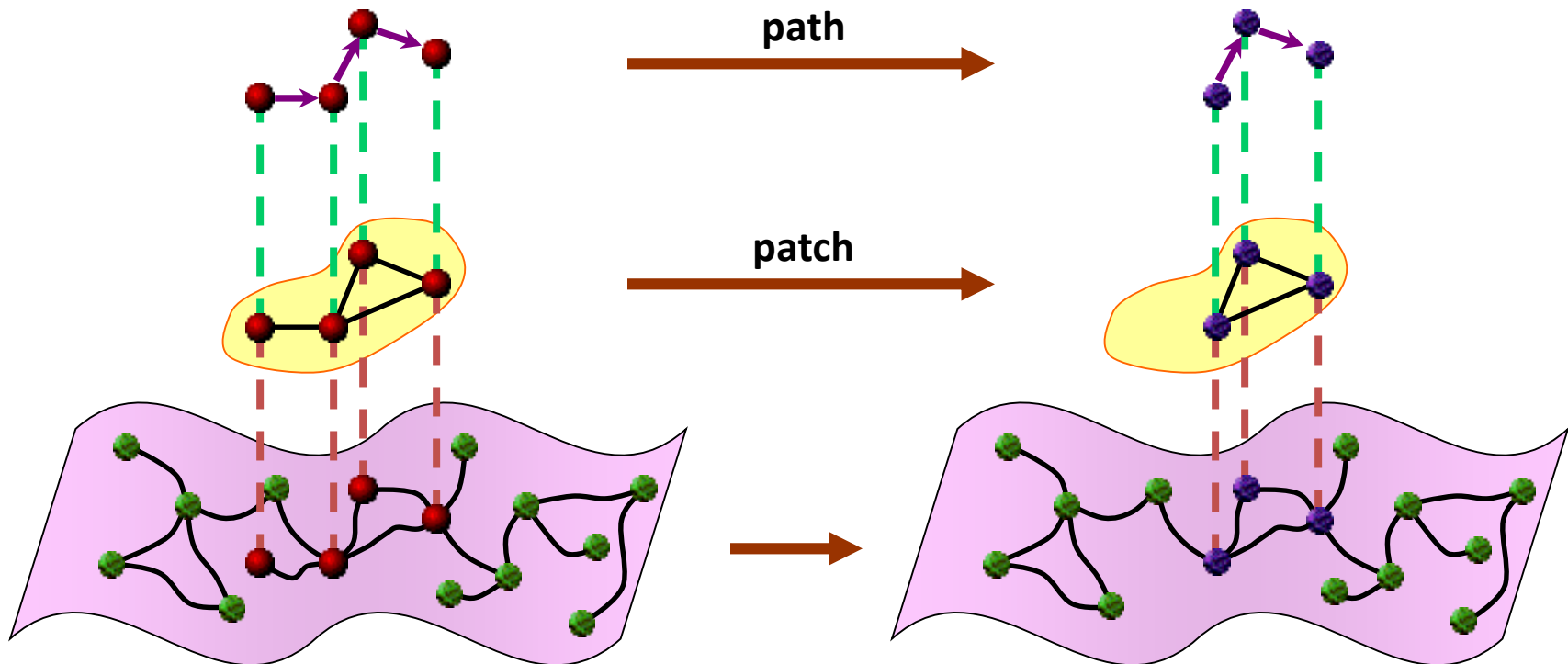


- MGS: a Formal Introduction
- Patch Transformations
- Differential Operators
- An Integrative Example: T-shape growth

# Patch Transformation

## ■ Motivations

- A straightforward implementation of the previous semantics
- Two pattern languages
  - Path patterns:  $p$ -neighborhood, close to regular expressions
  - Patch patterns: face/coface relation, arbitrary in dimension



# Patch: Syntax

## ■ Syntax: building collections

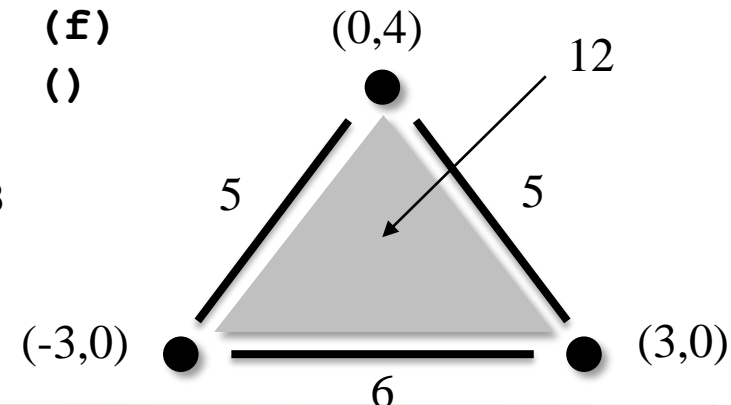
- Creation of a fresh cell

`new_cell dim faces cofaces`

- Binder `letcell ... in ...` & labeling \*

```
letcell v1 = new_cell 0 ()      (e1,e3)
and      v2 = new_cell 0 ()      (e1,e2)
and      v3 = new_cell 0 ()      (e2,e3)
and      e1 = new_cell 1 (v1,v2) (f)
and      e2 = new_cell 1 (v2,v3) (f)
and      e3 = new_cell 1 (v1,v3) (f)
and      f  = new_cell 2 (e1,e2,e3) ()
in
```

```
(0,4)*v1 + (3,0)*v2 + (-3,0)*v3
+ 5*e1 + 6*e2 + 5*e3
+ 12*f
```





# Patch: Syntax

## ■ Syntax: patterns

$$\begin{aligned} pat &::= pat \text{ } op \text{ } pat \mid clause \\ clause &::= (\sim)?x(:[dim = exp])? \\ op &::= < \mid > \mid \varepsilon \end{aligned}$$

- Pattern variable  $x$  corresponds to a collection element  $X.x$ 
  - In expressions  $exp$ ,  $x$  denotes  $X \in V^{var}$
  - In expressions  $exp$ ,  $\wedge x$  denotes  $x \in S^{var}$
- Tilded pattern variable  $\sim x$ 

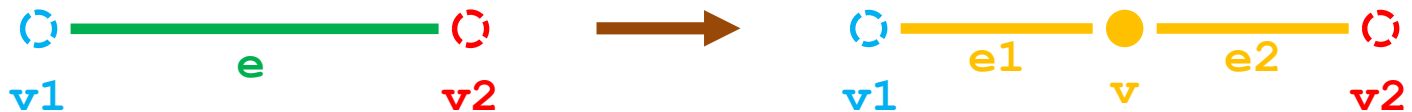
The element is matched but not consumed (can be matched by another rule)
- $x:[dim = exp]$  specifies the dimension of the matched element

The expression has to evaluate an integer
- $x < y$  means that  $\wedge x$  is a face of  $\wedge y$

# Patch: Vertex Insertion

## ■ Example

Splitting an edge by insertion of a vertex

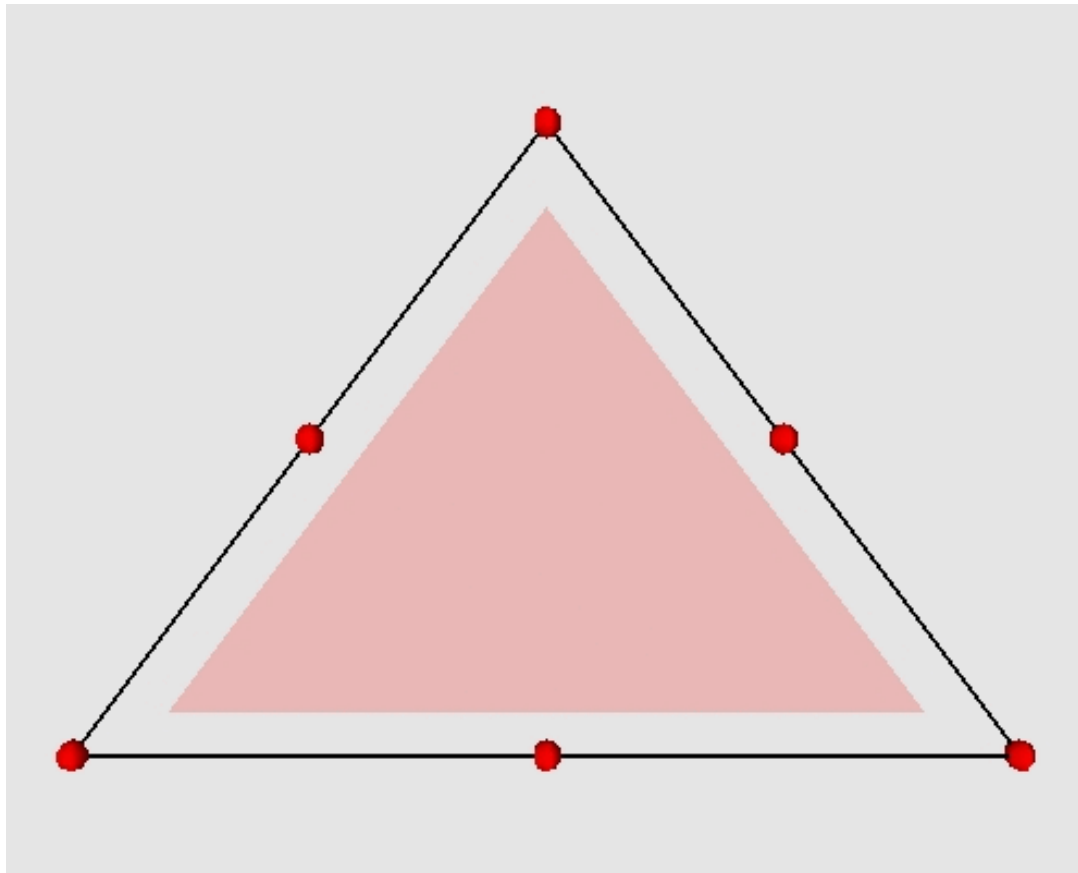


```
patch insert_vertex = {  
  ~v1 < e: [ dim = 1 ] > ~v2  
  =>  
    letcell v = new_cell 0 ()  
    and e1 = new_cell 1 (^v1, v) (cofaces ^e)  
    and e2 = new_cell 1 (^v2, v) (cofaces ^e)  
    in  
      (some expression) * v  
}
```

# Patch: Vertex Insertion

## ■ Example

Splitting an edge by insertion of a vertex

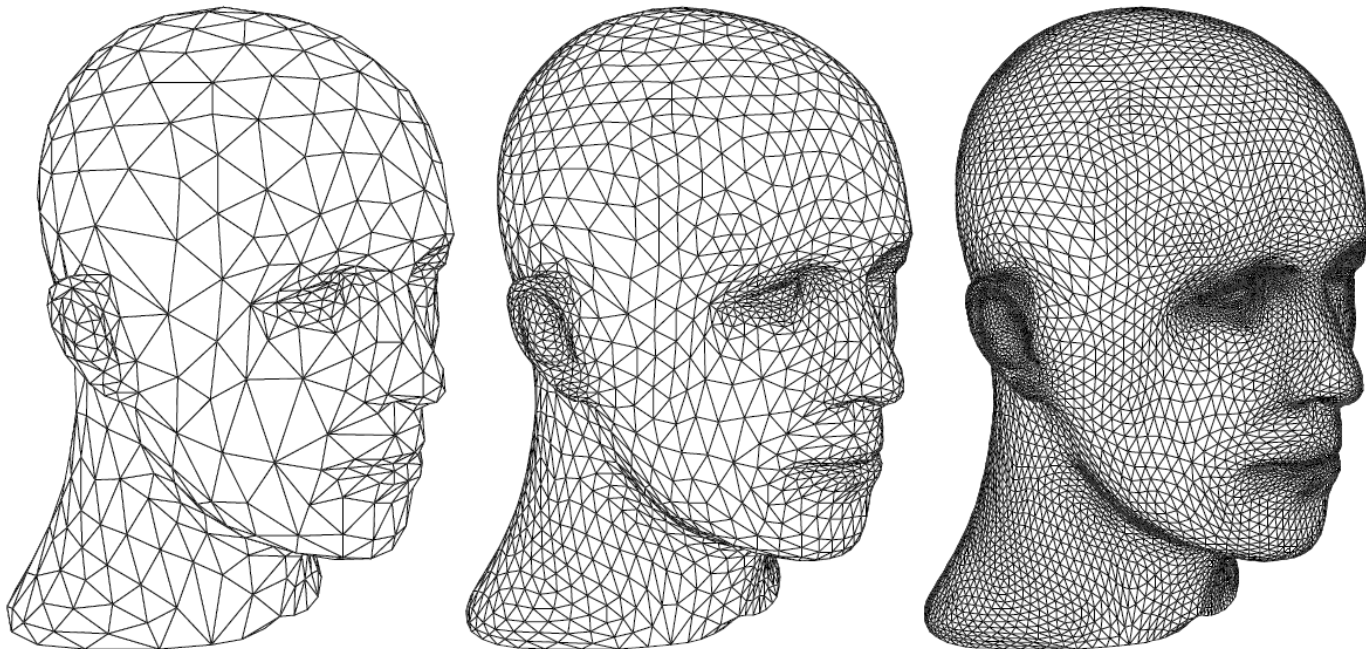


# Patch: Mesh Subdivision

## ■ Mesh subdivision

### □ Definition

“ *Subdivision defines a smooth curve or surface as the limit of a sequence of successive refinements* ”

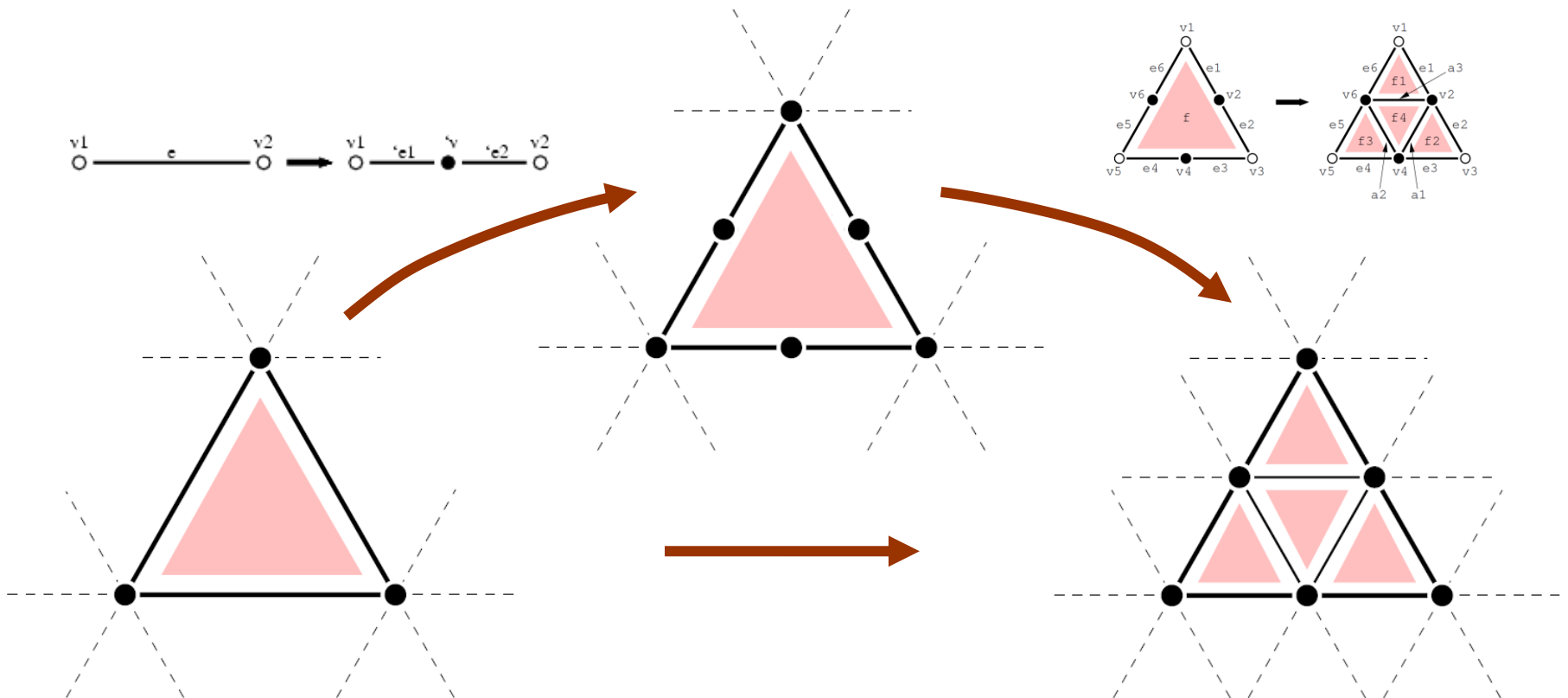


SIGGRAPH 98 Course Notes

# Patch: Mesh Subdivision

## ■ Polyhedral subdivision

- Inserting vertices on edges
- Splitting each hexagonal surface



# Patch: Mesh Subdivision

## ■ MGS Implementation

```
patch insert_vertex = { ... }
```

```
patch subdivide_face = {
```

```
  f: [ dim = 2 ]
```

```
  ~v1 < ~e1 < f > ~e1 > ~v2 < ~e2 < f > ~e2 >
```

```
  ~v3 < ~e3 < f > ~e3 > ~v4 < ~e4 < f > ~e6 >
```

```
  ~v5 < ~e5 < f > ~e5 > ~v6 < ~e6 < f > ~e4 > ~v1
```

```
  =>
```

```
    letcell a1 = new_cell 1 (^v2, ^v4)      (f1, f4)
```

```
    and      a2 = new_cell 1 (^v4, ^v6)      (f2, f4)
```

```
    and      a3 = new_cell 1 (^v6, ^v2)      (f3, f4)
```

```
    and      f1 = new_cell 2 (a1, ^e2, ^e3)  ()
```

```
    and      f2 = new_cell 2 (a2, ^e4, ^e5)  ()
```

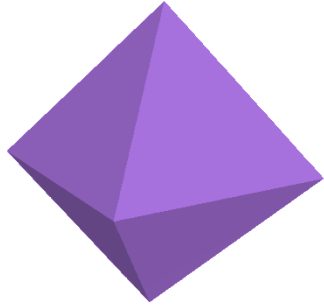
```
    and      f3 = new_cell 2 (a3, ^e6, ^e1)  ()
```

```
    and      f4 = new_cell 2 (a1, a2, a3)    () in
```

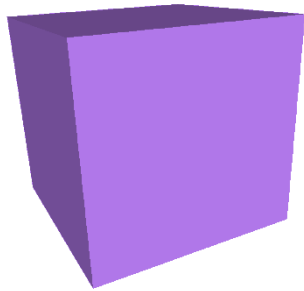
```
      `edge * a1 + ... + `triangle * f4
```

```
}
```

# Patch: Mesh Subdivision

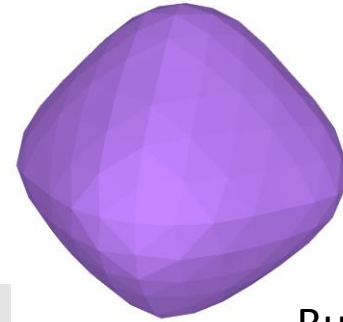


Triangular mesh

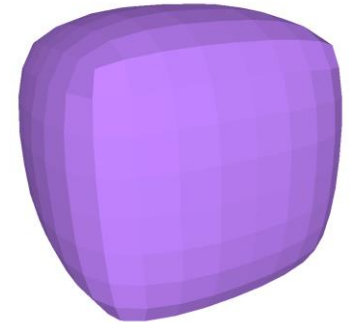
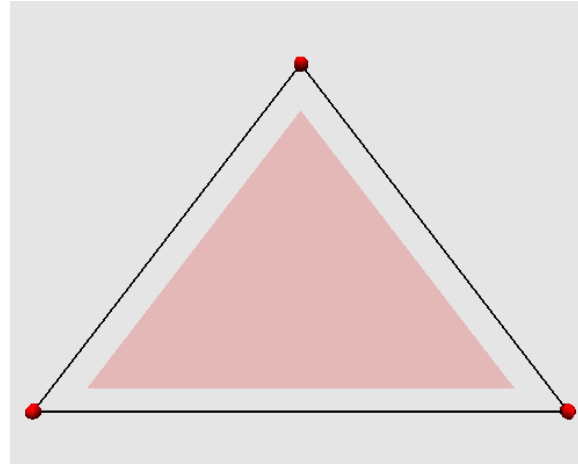


Quadrangular mesh

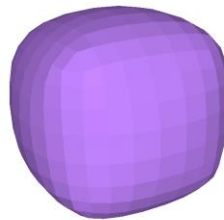
Loop



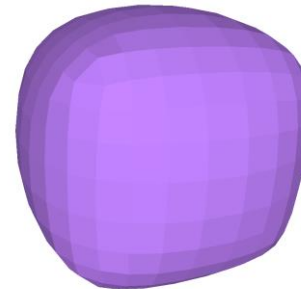
Butterfly



Kobbelt



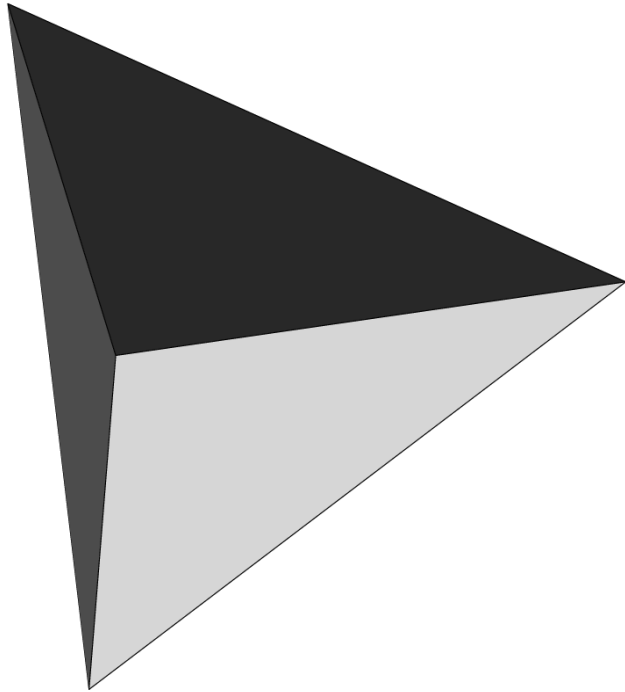
Catmull-Clark



Doo-Sabin

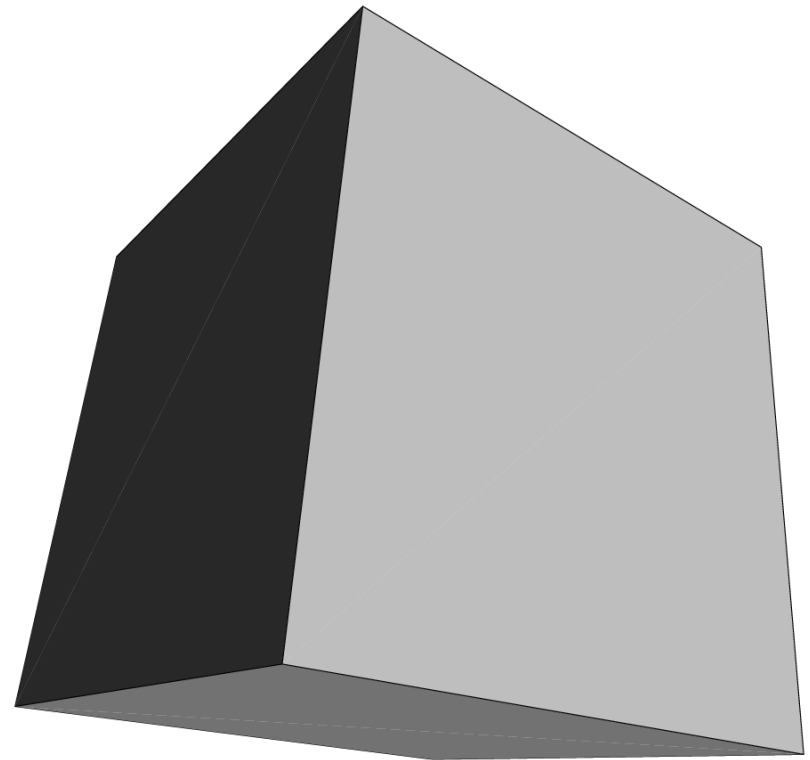
# Patch: Fractal

---



Sierpinsky Sponge (4 steps)

Menger Sponge (2 steps)





# Outline



- MGS: a Formal Introduction
- Patch Transformations
- Differential Operators
- An Integrative Example: T-shape growth

# Formalism Summary

## ■ Elements of Algebraic Topology

- Abstract Cellular Complex  $\mathcal{K} = (S, <)$
- ...
- ...

## ■ Topological Collections

- Formal Sums Representation  $c \in C_S(V) \Rightarrow c = \sum v_\sigma \cdot \sigma$
- Shape, Support  $\text{Shape}(c), |c|$
- Sub-collection, Merge  $s \subset c, c \uplus c'$

## ■ Transformation

- Collection Patterns/Expressions  $\alpha \in C_{S^{\text{var}}}(V^{\text{var}}), \beta \in C_{S \cup S^{\text{var}}}(\Sigma)$
- Rewriting Rules  $r = \alpha \Rightarrow \beta$
- Topological Rewriting  $c | \triangleright_R c'$  where  $R$  is a set of rules

# Diff. MGS: Algebraic Topology

## ■ Topological Chains

### □ Definition

Let  $\mathcal{K}$  be an ACC and let  $G$  be an *abelian group*.

A **topological chain** over  $\mathcal{K}$  with values in  $G$  is a function *null almost everywhere* from  $\mathcal{K}$  to  $G$ .  $C_{\mathcal{K}}(G)$  denotes the topological chains over  $\mathcal{K}$  with values in  $G$ .

### □ Motivations (*homology*)

Extends ACC with an algebraic structure

### □ Comparison with topological collections

- Similar to *collections with values in a group*

Main difference: chains are total functions

- Richer *algebraic structure*

$C_{\mathcal{K}}(G)$  has an abelian group structure

# Diff. MGS: Algebraic Topology

## ■ Topological Cochains

### □ Definition

Let  $\mathcal{K}$  be an ACC and let  $G$  and  $H$  be abelian groups.

The **topological cochains** of chains of  $C_{\mathcal{K}}(G)$  into  $H$  are group homomorphisms from  $C_{\mathcal{K}}(G)$  to  $H$ .

$C^{\mathcal{K}}(G, H)$  denotes the group of topological cochains from  $C_{\mathcal{K}}(G)$  to  $H$ .

### □ Representation with formal sums

$$T = \sum_{\tau \in \mathcal{K}} f \cdot \tau \text{ where } f \text{ are homomorphisms of } \text{Hom}(G, H)$$

### □ Application of a cochain on a chain

$$[T, c] = [\sum_{\tau \in \mathcal{K}} f \cdot \tau, \sum_{\sigma \in \mathcal{K}} v_{\sigma} \cdot \sigma] = \sum_{\omega \in \mathcal{K}} f^{\tau}(v_{\sigma})$$

$$\left[ \begin{array}{c} f_1 \\ \bullet \end{array} \xrightarrow{f_3} \begin{array}{c} f_2 \\ \bullet \end{array}, \begin{array}{c} v_1 \\ \bullet \end{array} \xrightarrow{v_3} \begin{array}{c} v_2 \\ \bullet \end{array} \right] = f_1(v_1) + f_2(v_2) + f_3(v_3)$$

# Diff. MGS: Summary

## ■ Elements of Algebraic Topology

- |                             |                                                                                                  |
|-----------------------------|--------------------------------------------------------------------------------------------------|
| □ Abstract Cellular Complex | $\mathcal{K} = (S, <)$                                                                           |
| □ Topological Chain         | $c \in C_{\mathcal{K}}(G) \Rightarrow c = \sum_{\sigma \in \mathcal{K}} v_{\sigma} \cdot \sigma$ |
| □ Topological Cochain       | $T \in C^{\mathcal{K}}(G, H) \Rightarrow T = \sum_{\tau \in \mathcal{K}} f^{\tau} \cdot \tau$    |

## ■ Topological Collections

- |                              |                                                             |
|------------------------------|-------------------------------------------------------------|
| □ Formal Sums Representation | $c \in C_S(V) \Rightarrow c = \sum v_{\sigma} \cdot \sigma$ |
| □ Shape, Support             | $\text{Shape}(c),  c $                                      |
| □ Sub-collection, Merge      | $s \subset c, c \uplus c'$                                  |

## ■ Transformation

- |                                   |                                                                                              |
|-----------------------------------|----------------------------------------------------------------------------------------------|
| □ Collection Patterns/Expressions | $\alpha \in C_{S^{\text{var}}}(V^{\text{var}}), \beta \in C_{S \cup S^{\text{var}}}(\Sigma)$ |
| □ Rewriting Rules                 | $r = \alpha \Rightarrow \beta$                                                               |
| □ Topological Rewriting           | $c \triangleright_R c'$ where $R$ is a set of rules                                          |

# Diff. MGS: Transformations vs. Cochains

## ■ Intersection Between Cochains and Transformations

□ Topological Cochain

$$T \in \mathcal{C}^{\mathcal{K}}(G, H) \Rightarrow T = \sum_{\tau \in \mathcal{K}} f^{\tau} \cdot \tau$$

□ Topological Rewriting

$$c \mid \triangleright_R c' \text{ where } R \text{ is a set of rules}$$

□ Rewriting Cochains

■ **Cochains** of  $T \in \mathcal{C}^{\mathcal{K}}(G, \mathcal{C}_{\mathcal{K}}(G)) = \text{Hom}(\mathcal{C}_{\mathcal{K}}(G), \mathcal{C}_{\mathcal{K}}(G))$

Mapping of topological chains to topological chains

$$\begin{array}{ccccccc} c & = & v_{\sigma_1} \cdot \sigma_1 & + & \dots & + & v_{\sigma_n} \cdot \sigma_n \\ \downarrow T & & \downarrow f_{\sigma_1} & & & & \downarrow f_{\sigma_n} \\ [T, c] & = & c_1 & \uplus & \dots & \uplus & c_n \end{array}$$

■ **Transformation** of the form  $R = \{ X.x \Rightarrow f^x(X) \}$

Application of a specific function on each cell of the collection

$$\mathbf{trans} \ T = \{ x \Rightarrow f^x(x) \}$$

■ One can show that

$$\forall c \in \mathcal{C}_{\mathcal{K}}(G) \quad c \mid \triangleright_R [T, c]$$

# Diff. MGS: Transport of Data

## ■ Algebraic handling of collection

- Usual functional map (when  $f^x(X)$  does not depend on  $x$ )
- Computing by moving data on the collection

when  $f^x(X)$  transports values from cells

- to their *p-neighbors* (i.e., the comma operator)

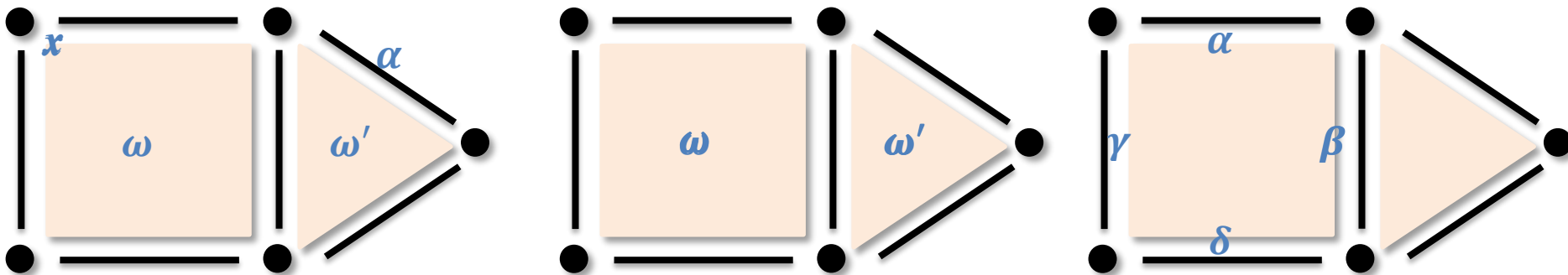
**trans** Eq1 = {  $x \Rightarrow \text{pNeighborsFold}(+, 0, x, p)$  }

- to their *faces* (i.e., the face operator)

**trans** Eq2 = {  $x \Rightarrow \text{CofacesFold}(+, 0, x)$  }

- to their *cofaces* (i.e., the coface operator)

**trans** Eq3 = {  $x \Rightarrow \text{FacesFold}(+, 0, x)$  }



# Differential Calculus in MGS

## ■ The boundary operator $\partial$

- Starting point of the elaboration of the *discrete differential calculus*

$$\partial \left[ \begin{array}{c} \text{triangle with vertices } 2, 7, -4 \text{ and red arrows} \end{array} \right] = \begin{array}{c} \text{triangle with vertices } 7+2, -2-4, 4-7 \text{ and blue dots} \end{array}$$

- *Coincides with Eq2* (transport of data to faces) with *orientation*

## ■ The derivative operator $\mathbf{d}$

- Defined w.r.t. *discrete Stokes' theorem*

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

$$\int_{\mathcal{D}} f(x) \mathbf{d}x = \int_{\mathcal{D}} \mathbf{d}F(x) = \int_{\partial \mathcal{D}} F(x)$$

Continuous Stokes' theorem

- *Coincides with Eq3* (transport of data to cofaces) with *orientation*



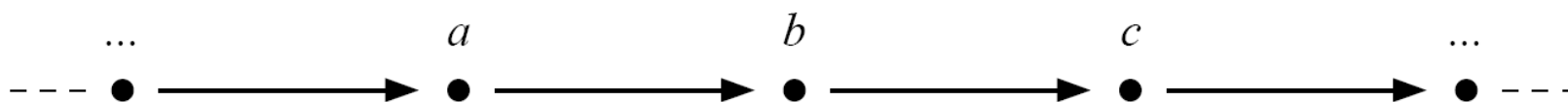
# Application: Generic Laplacian Operator

## ■ Laplacian in Discrete Differential Calculus

- Definition in terms of  $\partial$  and  $\mathbf{d}$

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

- Data transports (in dimension 1)



# Application: Generic Laplacian Operator

## ■ Laplacian in Discrete Differential Calculus

- Definition in terms of  $\partial$  and  $\mathbf{d}$

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

- MGS Implementation

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

# Application: Generic Laplacian Operator

## ■ Generic Implantation of a Diffusion Operator

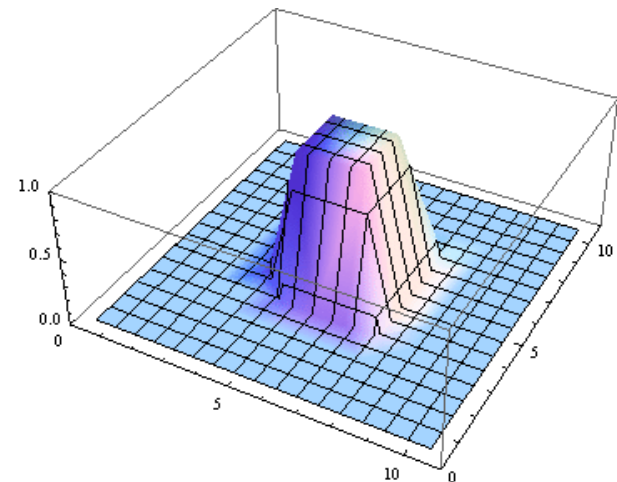
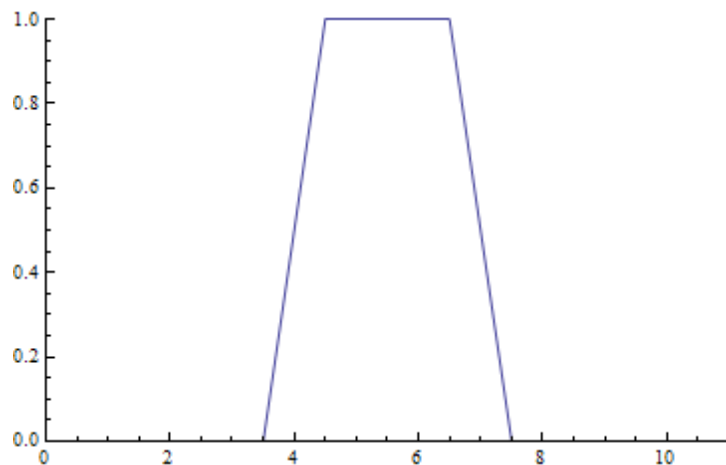
- Differential Equation & MGS implantation

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

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

- Continuous Simulations

The same operator works in any dimensions (here 1D and 2D)

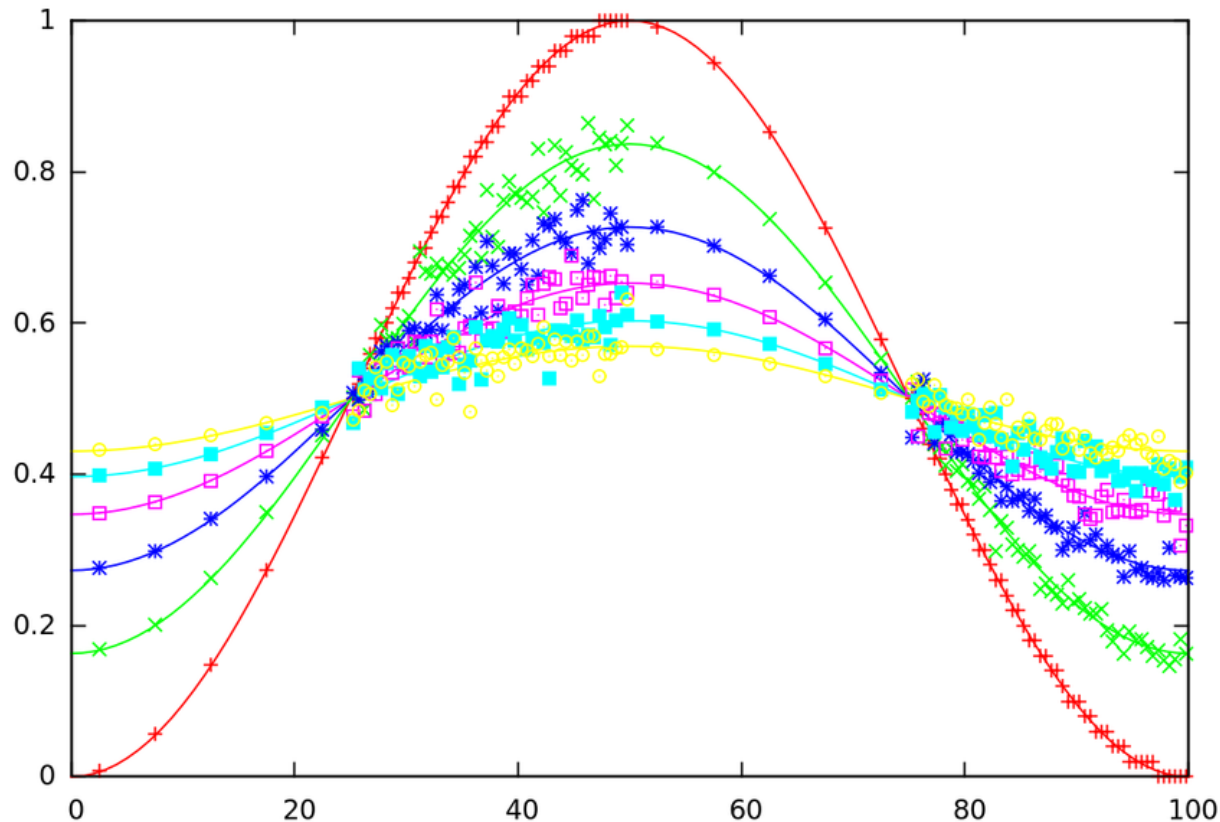


# Application: Generic Laplacian Operator

## ■ Generic Implantation of a Diffusion Operator

### □ Stochastic Simulations

Using another group  $G$  leads to random walk specifications



# Application: Generic Laplacian Operator

## ■ Generic Implantation of a Wave Operator

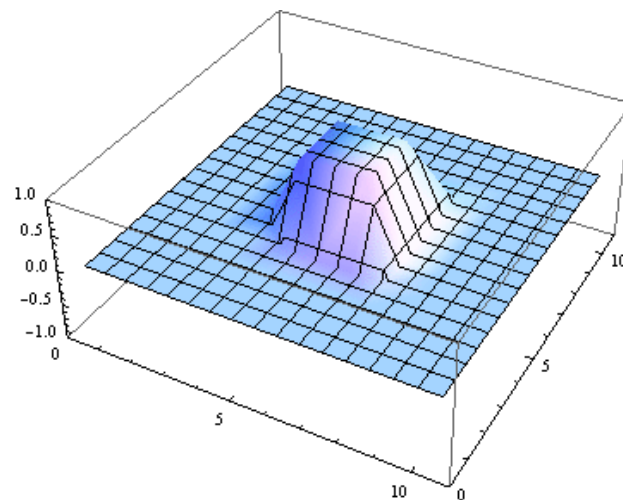
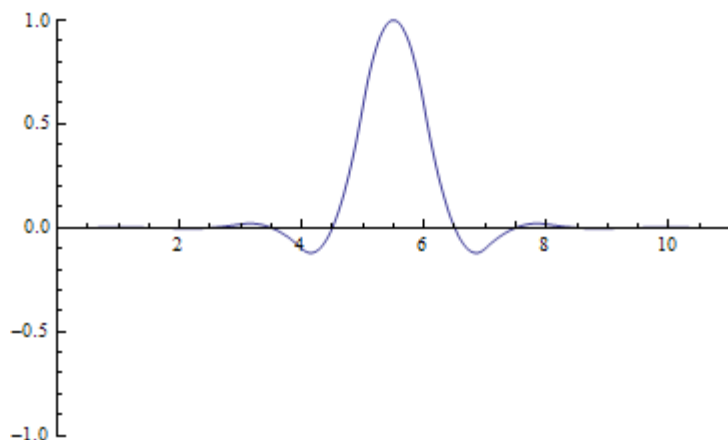
### □ Differential Equation & MGS implantation

$$\frac{\partial^2 u}{\partial t^2} = C \Delta u$$

```
fun wave[D,orient](u, u') =  
  let du' = C*Laplacian[orient=orient](Id)(u) in  
    (u+u'+du', u'+du')
```

### □ Continuous Simulations

The same operator works in any dimensions (here 1D and 2D)

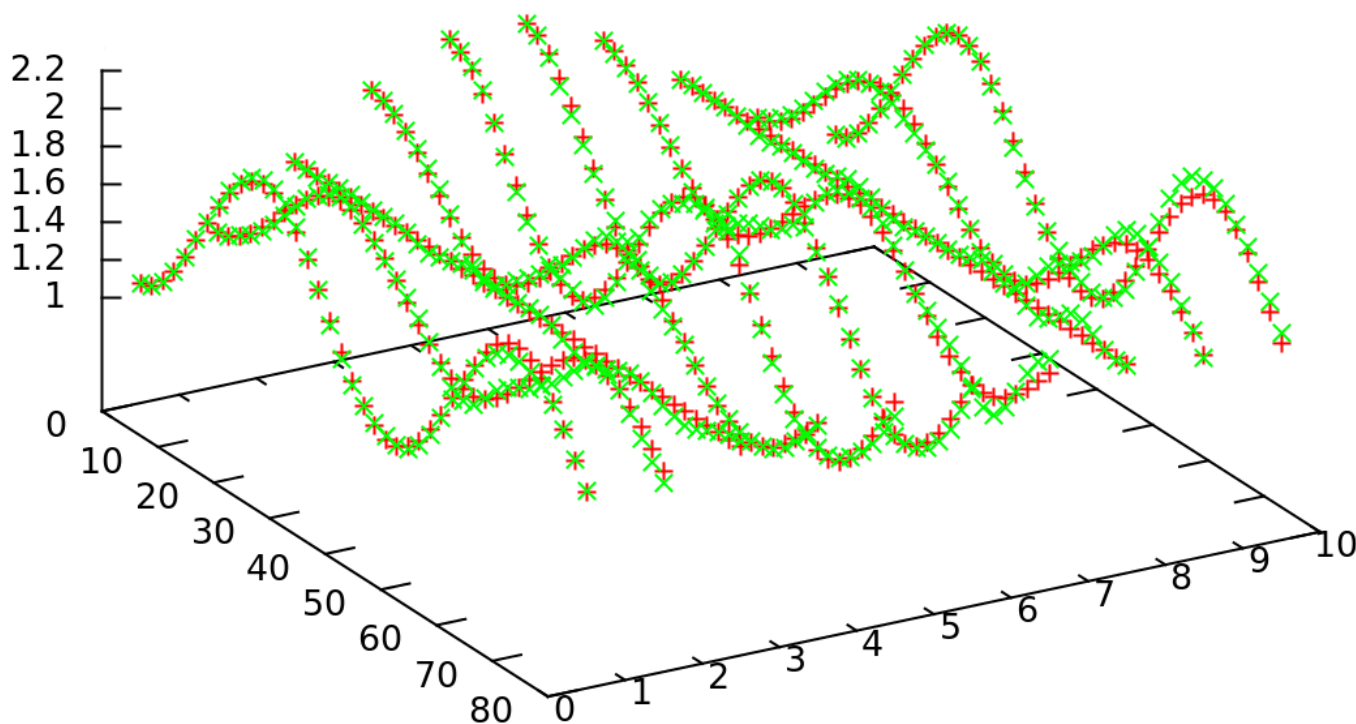


# Application: Generic Laplacian Operator

## ■ Generic Implantation of a Wave Operator

### □ Stochastic Simulations

Using another group  $G$  leads to random walk specifications



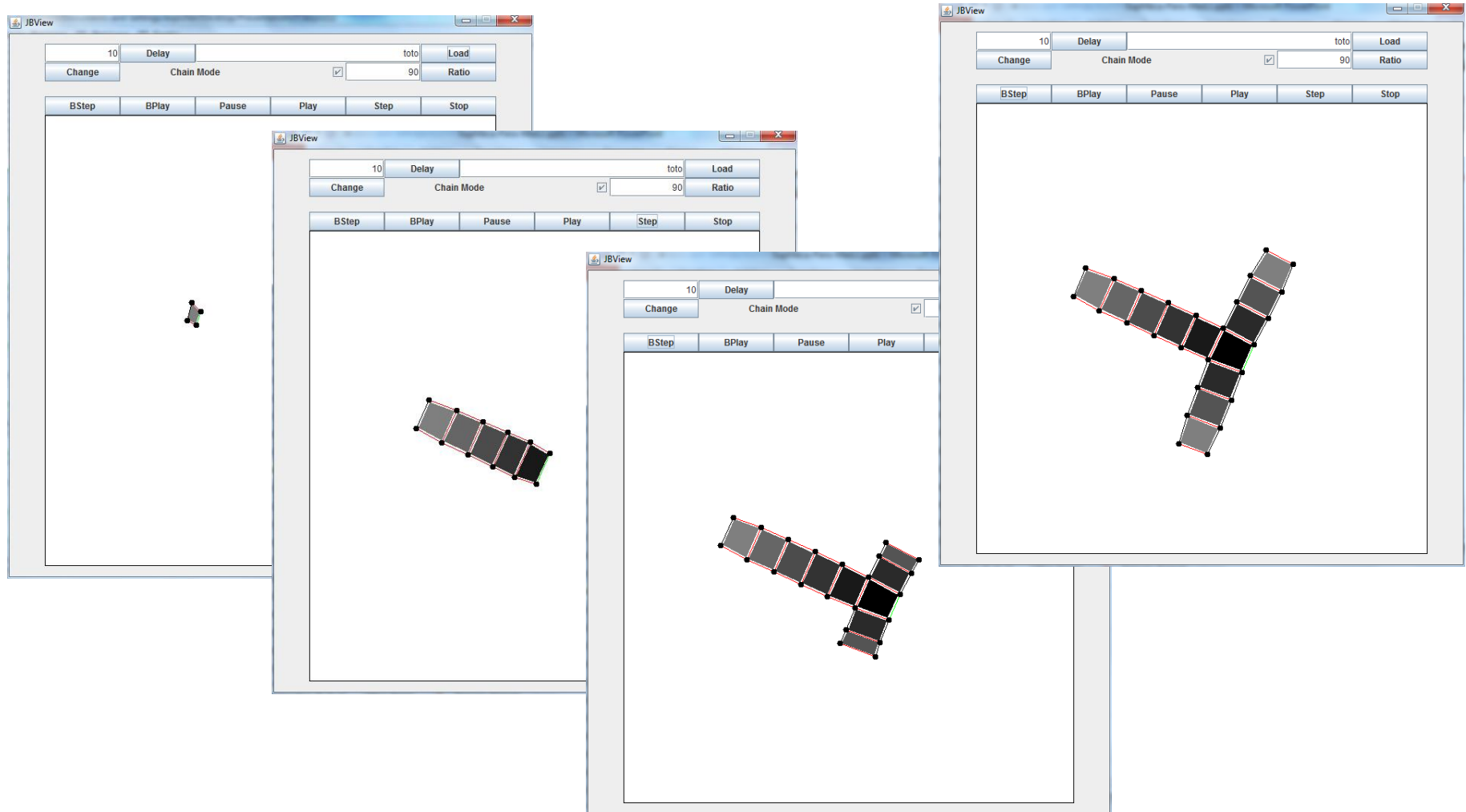
# Outline



- MGS: a Formal Introduction
- Patch Transformations
- Differential Operators
- An Integrative Example: T-Shape Growth

# T-Shape Growth

## ■ Spatial Programming Classical Example





# T-Shape Growth

## ■ Differential Transformations for Spring Forces

### □ Elastic Stress

$$\vec{F} = \nabla \cdot \sigma(p)$$

### □ MGS Implantation (p-neighbors data transport)

```
trans ElasticStress[k=1.0,L0=5.0,dt=0.1] = {  
  
  x => pneighborsfold(  
    (fun y F -> (  
      let d = distance(x,y) in  
      let stress = k * (d - L0) / d in  
      F + stress * (y-x)  
    ), F_null, x)  
  
}
```

# T-Shape Growth

## ■ Patch Transformations for Cells Divisions

### □ MGS Implantation

```
patch CellsDivision = {
```

```
  ~v1 < e12 < ~f:[dim=2] > e12 > ~v2
  when (e12 == `Apical) => (
    letcell v3(0)
    and v4(0)
    and e23(1, (^v2, v3))
    and e34(1, (v3, v4))
    and e41(1, (v4, ^v1))
    and nf(2, (^e12,e23,e34,e41)) in
      ( v2 + 0.05 * (v2-f) ) * v3 +
      ( v1 + 0.05 * (v1-f) ) * v4 +
      `Internal * ^e12 + `Lateral * e23 +
      `Apical * e34 + `Lateral * e41 + (NextFGP(f)) * nf
  }
```