Spatial Computing in MGS

Lecture II – MGS & Applications

Antoine Spicher[1] & Olivier Michel[1] & Jean-Louis Giavitto[2]

mgs.spatial-computing.org/
www.spatial-computing.org/mgs:tutorial

[1]LACL, University Paris Est Créteil

[2]IRCAM, CNRS – UPMC

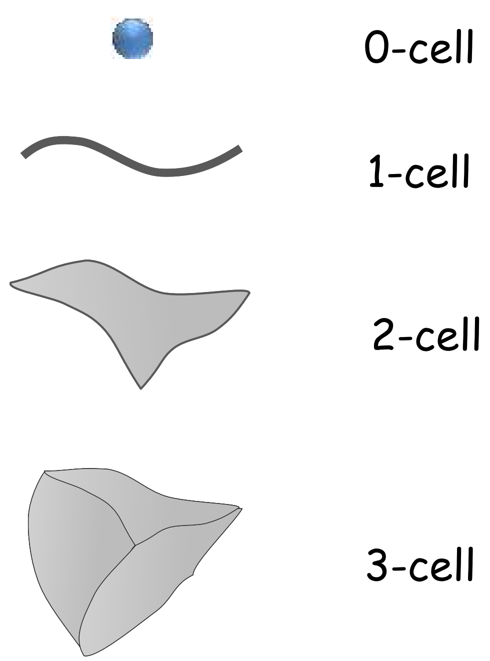
UCNC’12 – Sept. 2012

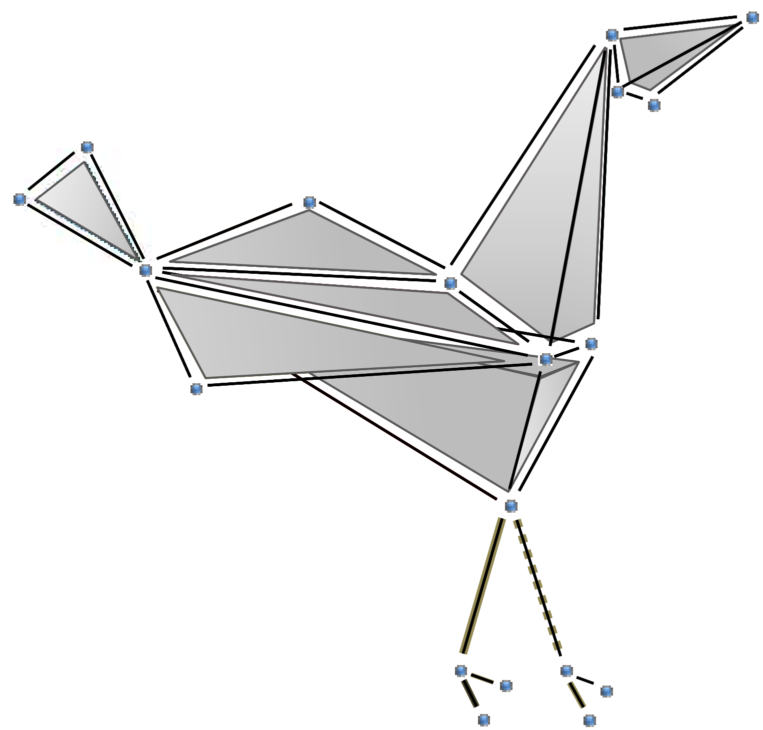
Outline

- 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*





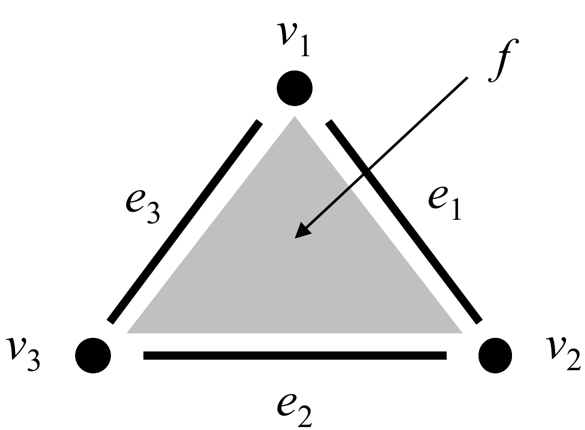
MGS Formalism: Collection

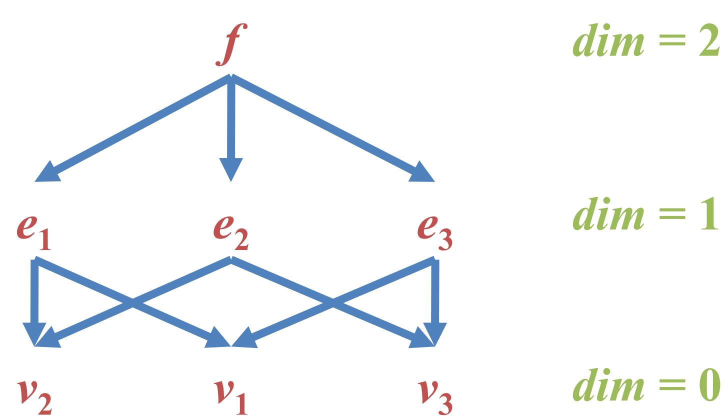
Abstract Cellular Complex (ACC)

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

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

- $S \subset \mathrm{S}$ *is a* **set of topological cells**,
- $\prec \, \subset S \times S$ *is a partial order on* S, *called the* **incidence relation**,
- *the* **dimension** *is* ***monotone*** *for* $\prec$: $\sigma_1 \prec \sigma_2 \Longrightarrow \dim(\sigma_1) < \dim(\sigma_2)$.





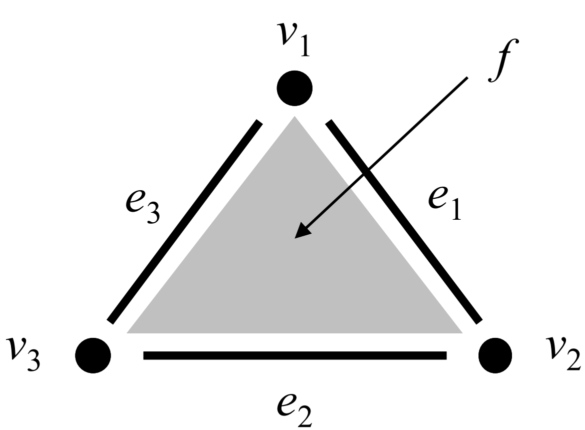
MGS Formalism: Collection

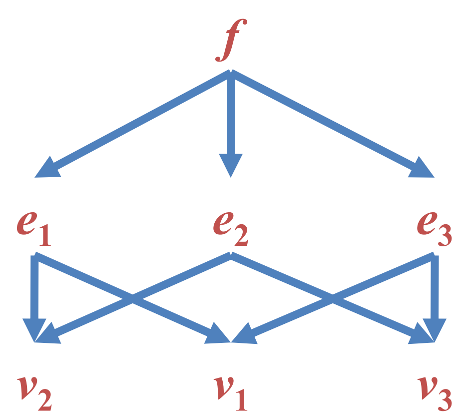
- Some neighborhoods on ACC
  - Face/Coface relationship (<, >)

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

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

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





$v_1 < e_1 > v_2$

$v_2 < e_2 > v_3$

$v_3 < e_3 > v_1$

$e_1 > f$

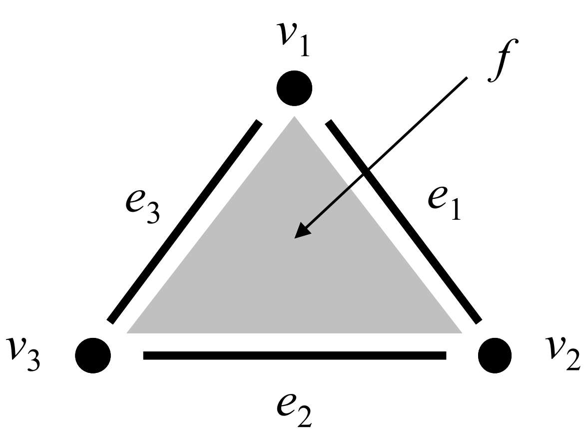
$e_2 > f$

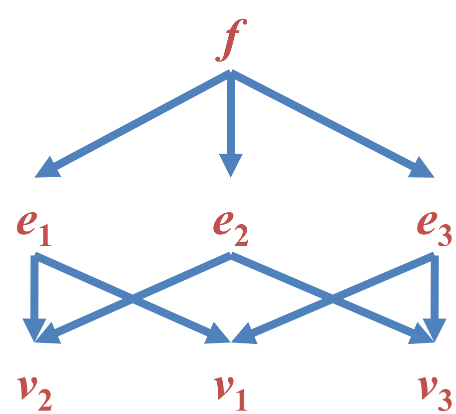
$e_3 > f$

MGS Formalism: Collection

- Some neighborhoods on ACC
  - Face/Coface relationship ($<, >$)
  - p-Neighborhood ($,_p$)

> *Let $\mathcal{K}$ be an ACC, τ_1 and τ_2 two n-cells of $\mathcal{K}$ and p an integer.*
>
> *The cells τ_1 and τ_2 are said **p-neighbors** if there exists $\sigma \in \mathcal{K}$ such that*
>
> - $\tau_1 \succ \sigma$ *and* $\tau_2 \succ \sigma$ *if* $n > p$, *or*
> - $\tau_1 \prec \sigma$ *and* $\tau_2 \prec \sigma$ *if* $n < p$
>
> *This relation is denoted by a comma:* $\tau_1 ,_p \tau_2$.





$v_1 ,_1 v_2$

$v_2 ,_2 v_3$

$e_3 ,_0 e_1$

$e_2 ,_2 v_1$

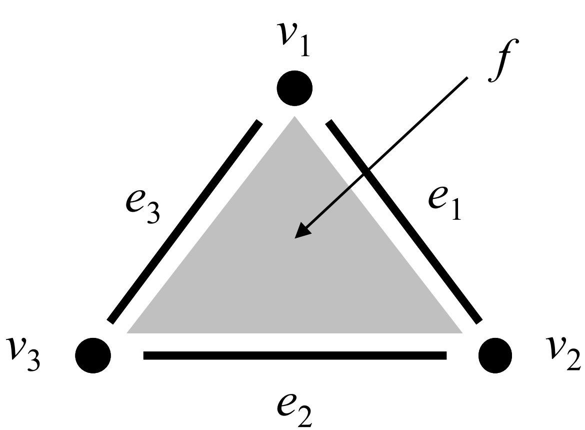
...

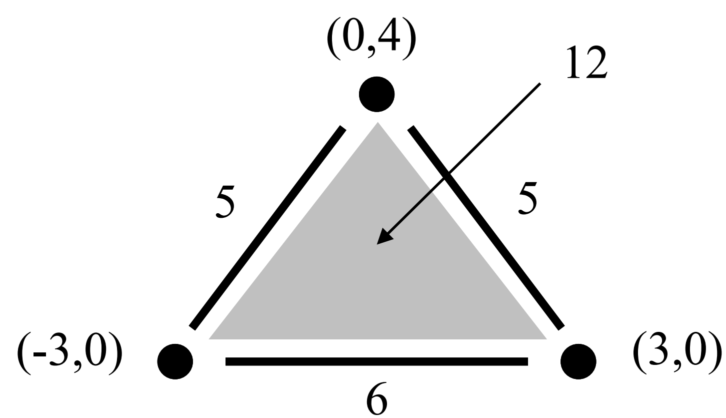
MGS Formalism: Collection

- 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**. $\mathcal{C}_S(\mathrm{V})$ *denotes the set of collections with values in* **V**.

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

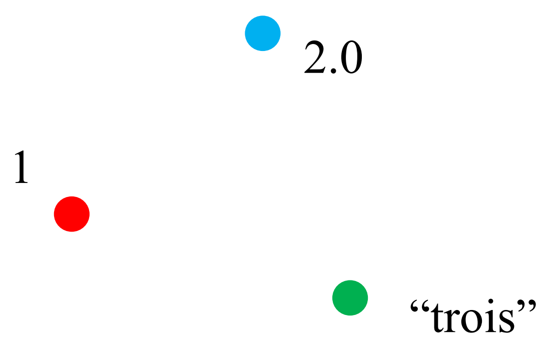




MGS Formalism: Collection

- 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

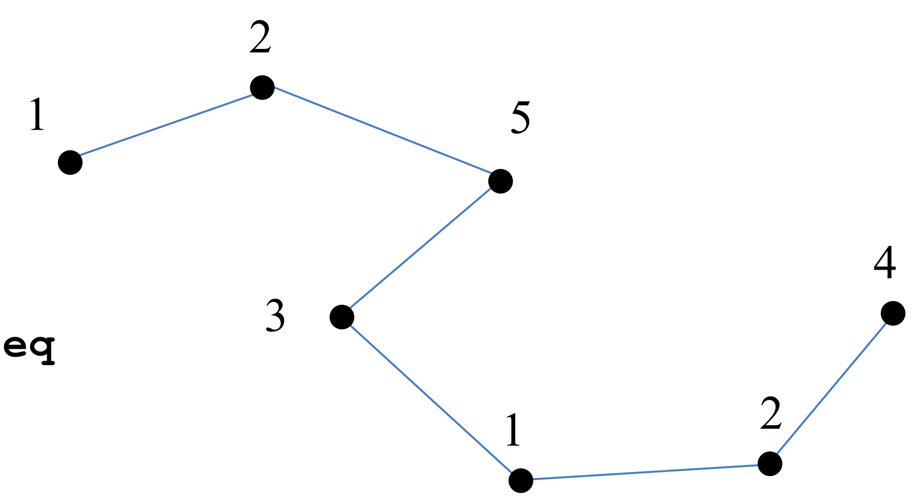
```
{ a = 1, b = 2.0, c = “trois” }
```



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

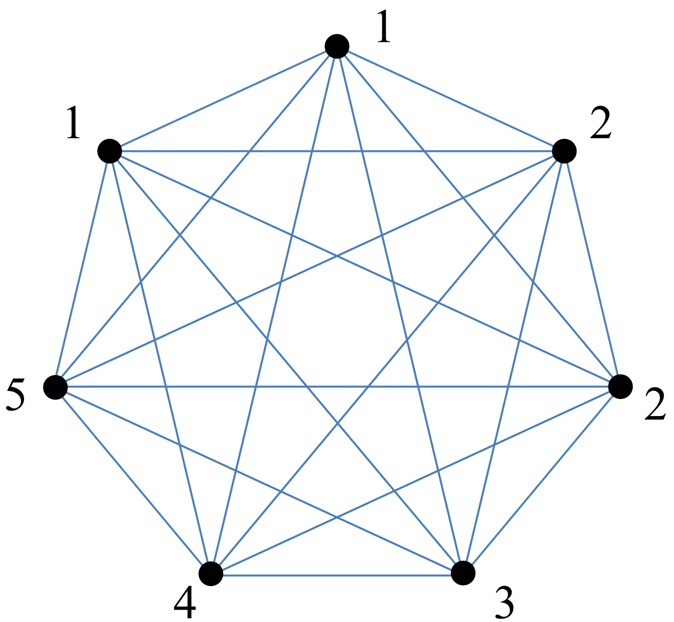
`(1, 2, 5, 3, 1, 2, 4):seq`



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

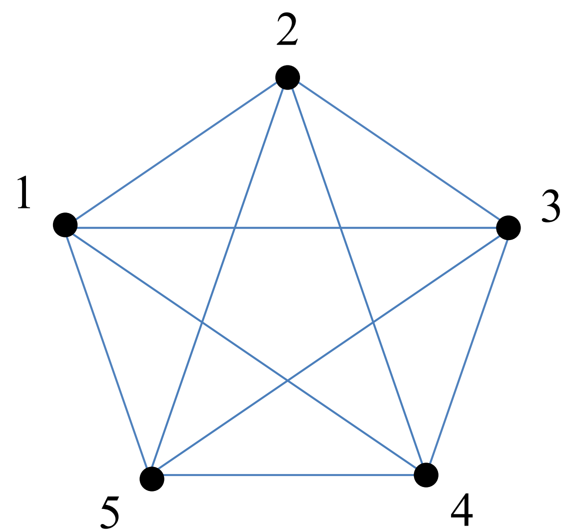
(1, 2, 5, 3, 1, 2, 4):bag



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
```

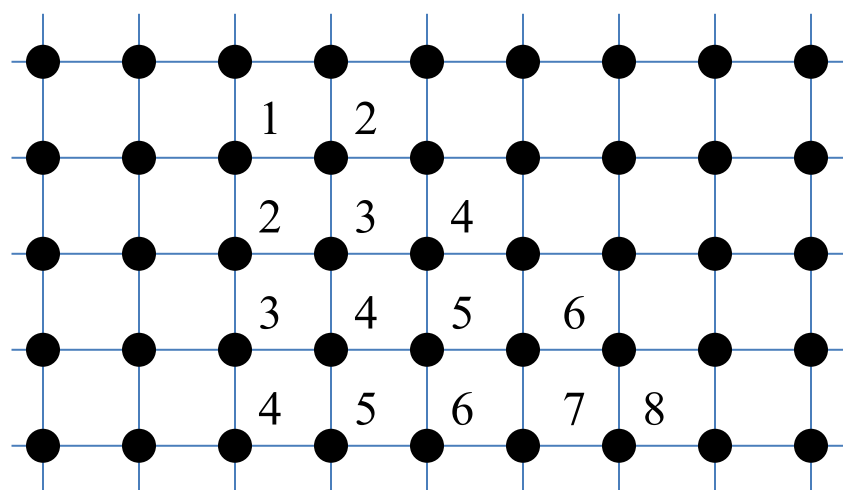


MGS Formalism: Collection

- Types of collections
  - Depending on the topology of the underlying cellular complex
  - GBF collections
    - Let $G = \langle d_1, d_2, \ldots \mid r_1, r_2, \ldots \rangle$ be a finitely generated group
    - $\mathcal{K} = (G, \{ (g ,_1 g \pm d_i) \mid g \in G \})$, the Cayley's graph of G

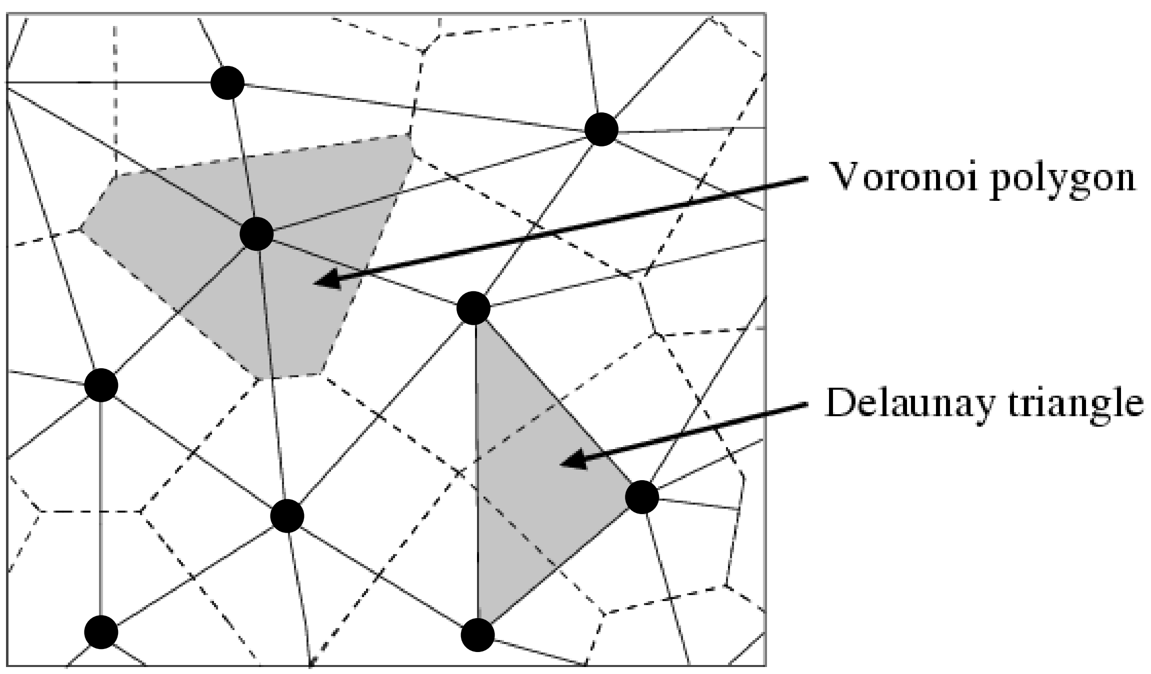
```
gbf NEWS = < north, east, west, south ;
             north = -south, east = -west > ;;

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



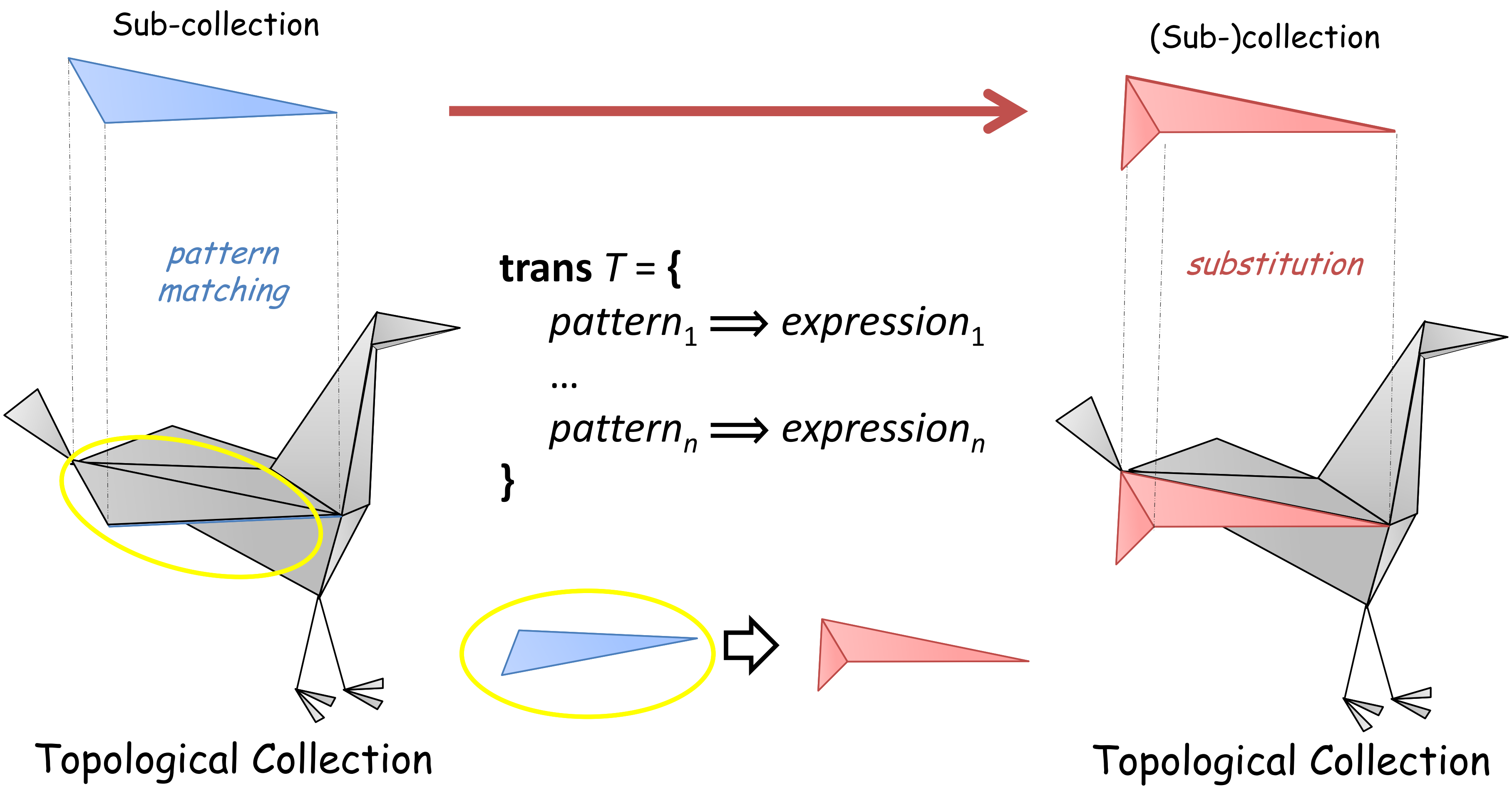
MGS Formalism: Collection

- 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



trans T = {

  $pattern_1 \Longrightarrow expression_1$

  …

  $pattern_n \Longrightarrow expression_n$

}

MGS Formalism: Transformation

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

$$pattern \Longrightarrow expression$$

  - Semantics of a transformation: ***topological rewriting***

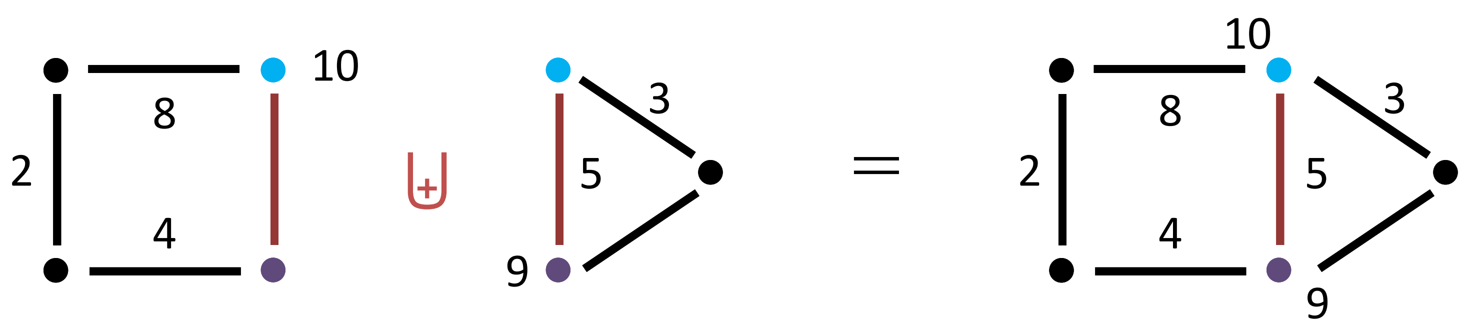
- Requirements
  - Topological collection patterns
  - Topological collection expressions, environments and evaluation
  - Pattern matching
  - Rewriting rule/relation

MGS Formalism: Transformation

- Some notations
  - Collection: $c = \sum v.\sigma$
  - Shape: $\mathrm{Shape}(c) = \mathcal{K}$
  - Support: $|c| = \{ \sigma \in \mathrm{Shape}(c) \mid c(\sigma) \text{ is defined } \}$
  - Extension: $c' = c_{|\mathcal{K}}$

    $c'(\sigma) = c(\sigma)$ when $\sigma \in \mathcal{K} \cap \mathrm{Shape}(c)$, and is undefined on $\mathcal{K}-\mathrm{Shape}(c)$
  - Merge: $c_1 \uplus c_2$

    $c_{1_{|\mathcal{K}}} + c_{2_{|\mathcal{K}}}$ where $\mathcal{K} = \mathrm{Shape}(c_1) \cup \mathrm{Shape}(c_2)$

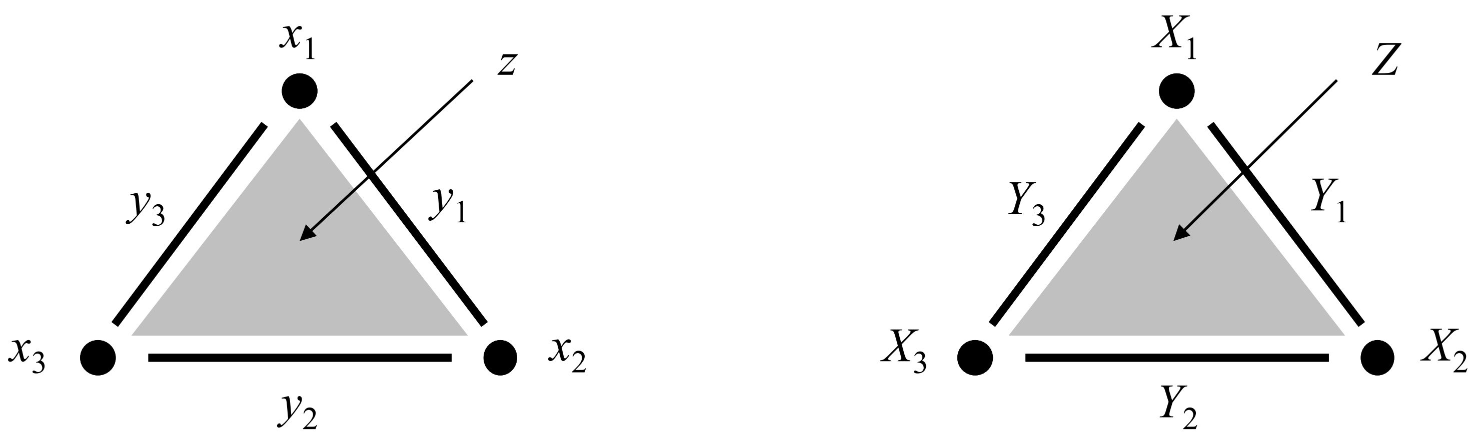


MGS Formalism: Transformation

- Topological Collection Patterns
  - Let consider the two sets of variables:
    - $S^{\text{var}} = \{ x_1, x_2, \ldots \}$ variables denoting cells

      Elements of are ranked by dimension (*i.e.*, $S^{\text{var}} = \bigcup_n S_n^{\text{var}}$)
    - $V^{\text{var}} = \{ X_1, X_2, \ldots \}$ 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 Σ
    - A ***collection expression*** is a collection of $\mathcal{C}_{\mathbf{S}^{\mathrm{var}} \cup \mathbf{S}}(\Sigma)$
  - Environments
    - $\Gamma_{\mathbf{S}} = \mathbf{S}^{\mathbf{var}} \longrightarrow \mathbf{S}$
    - $\Gamma_{\mathbf{V}} = \mathbf{V}^{\mathbf{var}} \longrightarrow \mathbf{V}$
  - Evaluation function

$$\zeta\colon\ \mathcal{C}_{\mathbf{S}^{\mathrm{var}} \cup \mathbf{S}}(\Sigma) \times \Gamma_{\mathbf{S}} \times \Gamma_{\mathbf{V}} \longrightarrow \mathcal{C}_{\mathbf{S}}(\mathbf{V})$$

MGS Formalism: Transformation

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

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

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

MGS Formalism: Transformation

- Rewriting rule & rewriting relation
  - Rewriting rule $\boldsymbol{\alpha} \Longrightarrow \boldsymbol{\beta}$
    - α is a topological collection pattern
    - β is a topological collection expression
  - One-step rewriting relation: $c_1 \rhd_{\alpha \Rightarrow \beta} c_2$ iff
    - $c_1 = \boldsymbol{l} \uplus \boldsymbol{c}$ ($\boldsymbol{l}$ is the redex and $\boldsymbol{c}$ is the context)
      such that $\boldsymbol{\alpha}$ matches $\boldsymbol{l}$ in c_1 with some environments $\boldsymbol{\rho_S}$ and $\boldsymbol{\rho_V}$
    - $c_2 = \boldsymbol{r} \uplus \boldsymbol{c}$
      such that $\boldsymbol{r} = \zeta(\boldsymbol{\beta}, \boldsymbol{\rho_S}, \boldsymbol{\rho_V})$
    - $(\mathrm{Shape}(\boldsymbol{r}) - \mathrm{Shape}(\boldsymbol{l})) - \mathrm{Shape}(\boldsymbol{c}) = \emptyset$
  - $\rhd_R$ trivial extension to a set R of rules

MGS Formalism: Transformation

- Topological Rewriting
  - $|\triangleright_R$ *parallel rewriting* of a set R of rules

$$
\begin{array}{ccccccccc}
c_1 & = & l_1 & \uplus & \ldots & \uplus & l_n & \uplus & c \\
\big\downarrow |\triangleright_R & & \big\downarrow \triangleright_R & & & & \big\downarrow \triangleright_R & & \big\downarrow \triangleright_R \\
c_2 & = & r_1 & \uplus & \ldots & \uplus & r_n & \uplus & c
\end{array}
$$

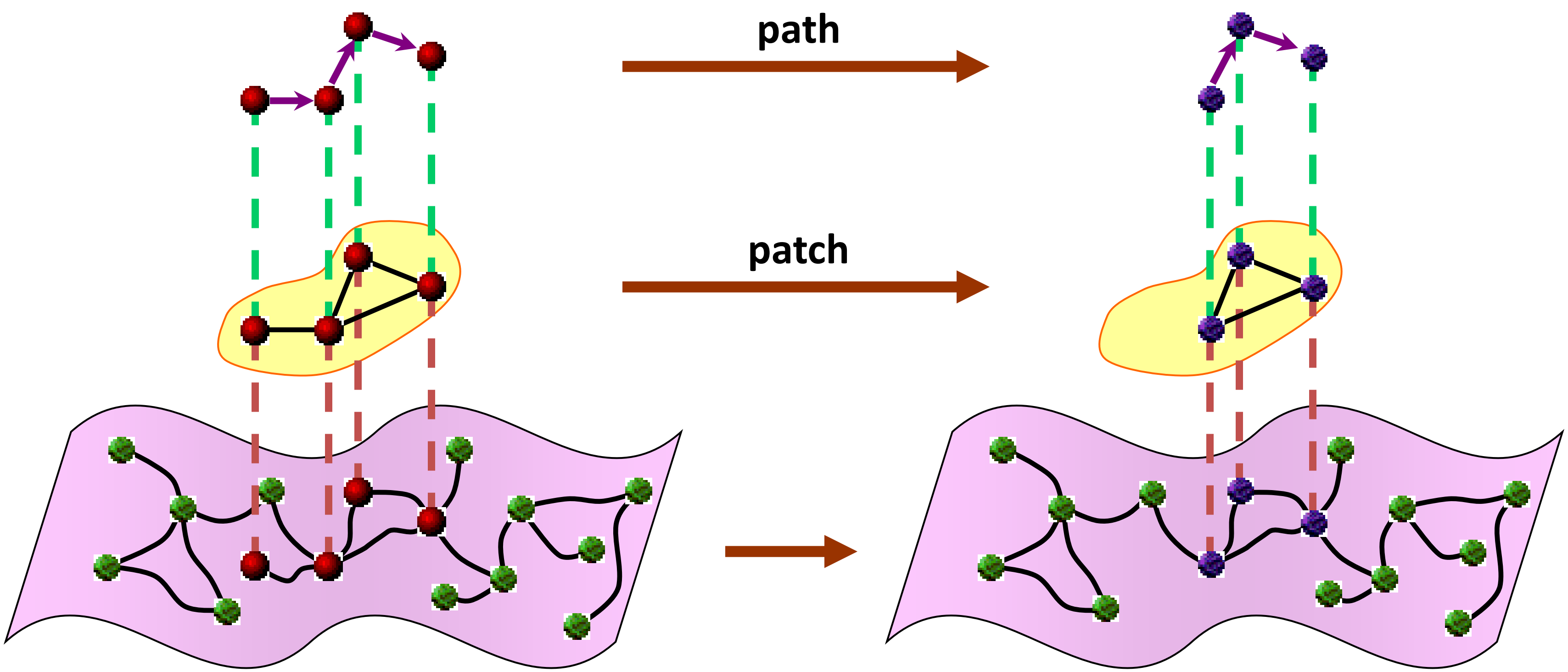
$$|l_i| \cap |l_j| = \emptyset \ \mathit{for\ all}\ i \neq j$$

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



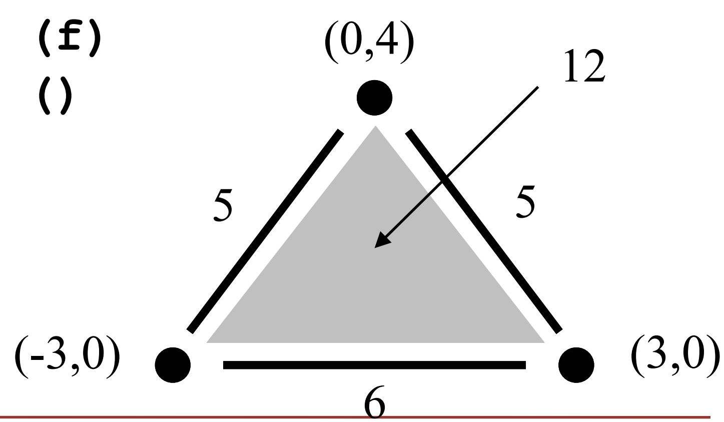
Patch: Syntax

- 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

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

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

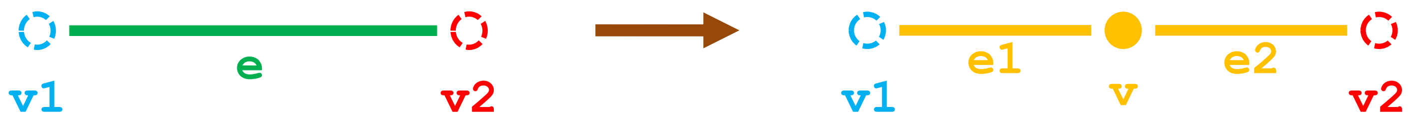
  The element is matched but not consumed (can be matched by another rule)
- `x:[dim = `*exp*`]` specifies the dimension of the matched element

  The expression has to evaluate an integer
- `x < y` means that `^x` is a face of `^y`

Patch: Vertex Insertion

- Example

  Splitting an edge by insertion of a vertex

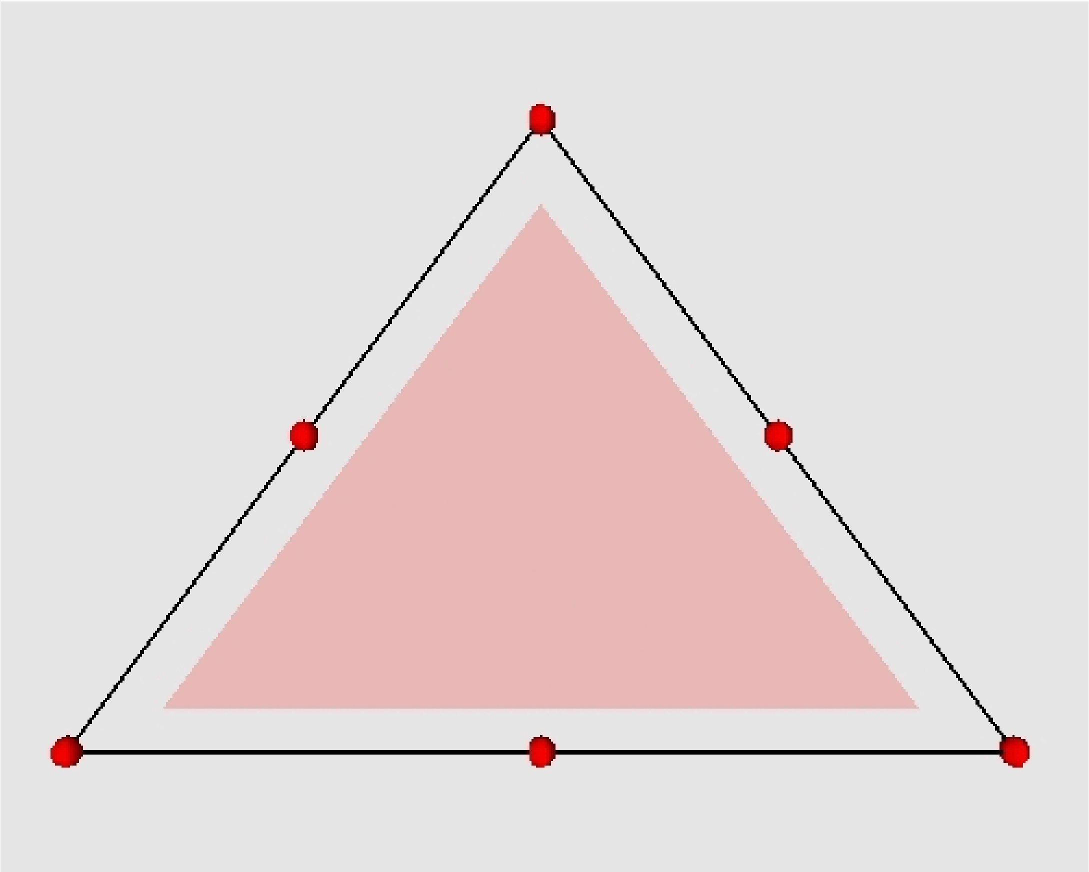



```
patch insert_vertex = {
  ~v1 < e:[ dim = 1 ] > ~v2
    =>
      letcell v  = new_cell 0 ()        ()
      and     e1 = new_cell 1 (^v1,v) (cofaces ^e)
      and     e2 = new_cell 1 (^v2,v) (cofaces ^e)
      in
        (some expression) * v
}
```

Patch: Vertex Insertion

- Example

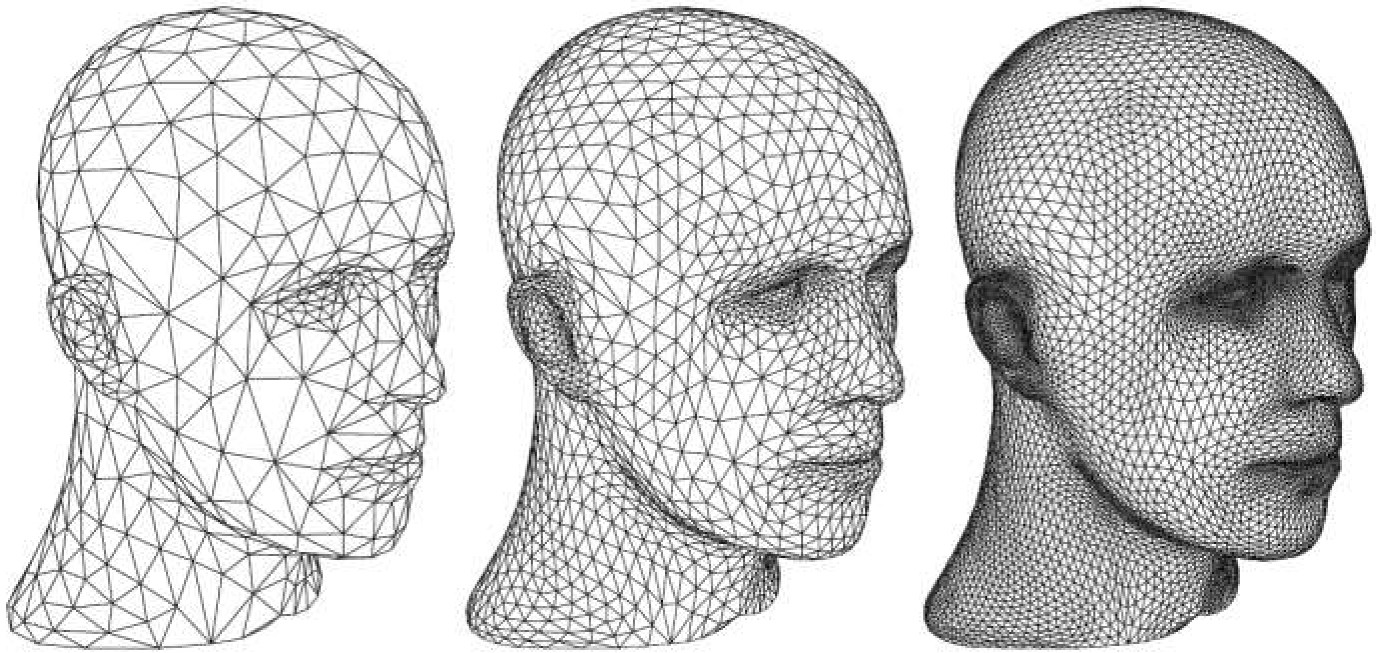
  Splitting an edge by insertion of a vertex



Patch: Mesh Subdivision

- Mesh subdivision
  - Definition

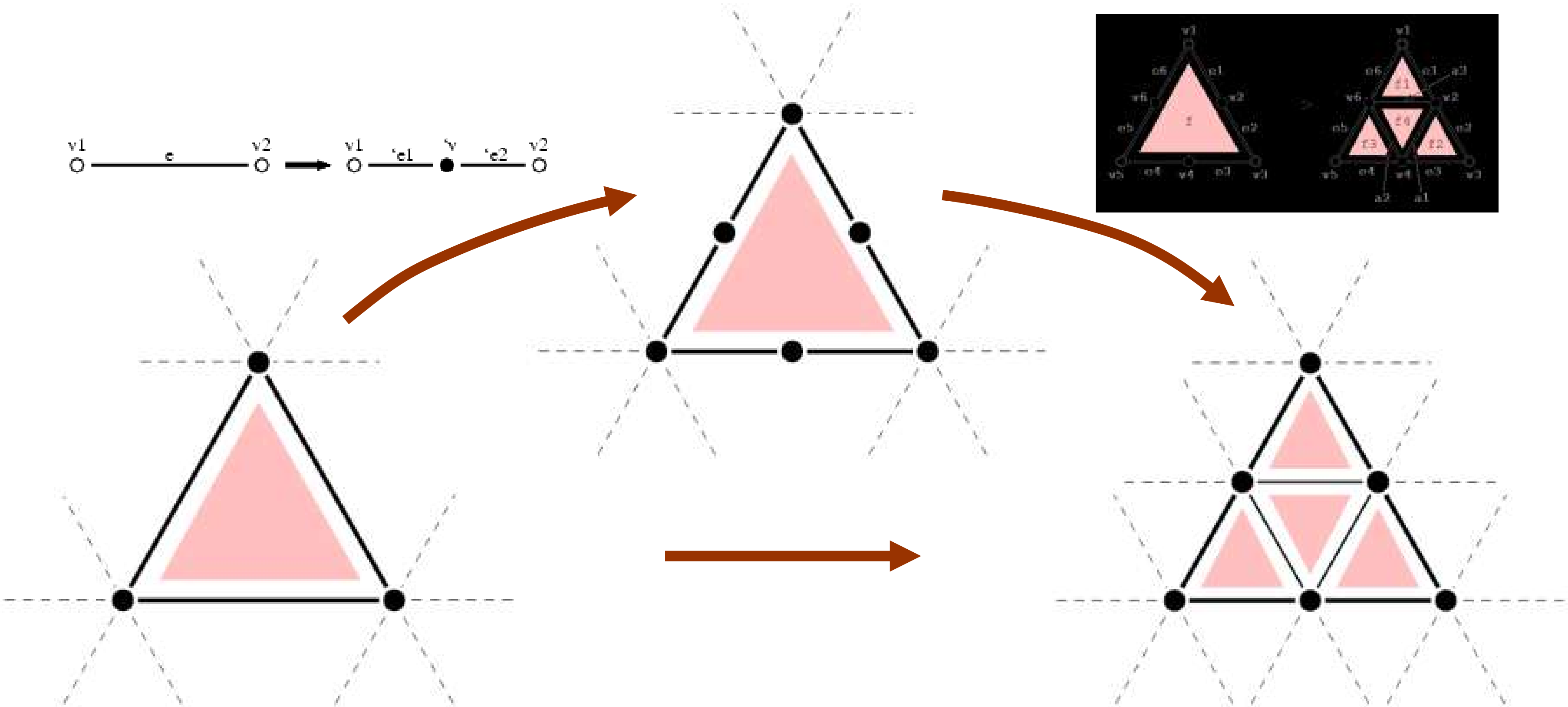
“ *Subdivision defines a smooth curve or surface as the limit of a sequence of successive refinements* ”



SIGGRAPH 98 Course Notes

Patch: Mesh Subdivision

- Polyhedral subdivision
  - Inserting vertices on edges
  - Splitting each hexagonal surface



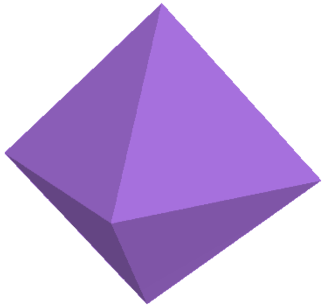
Patch: Mesh Subdivision

- MGS Implementation

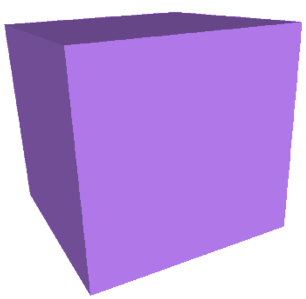
```
patch insert_vertex = { … }

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
}
```

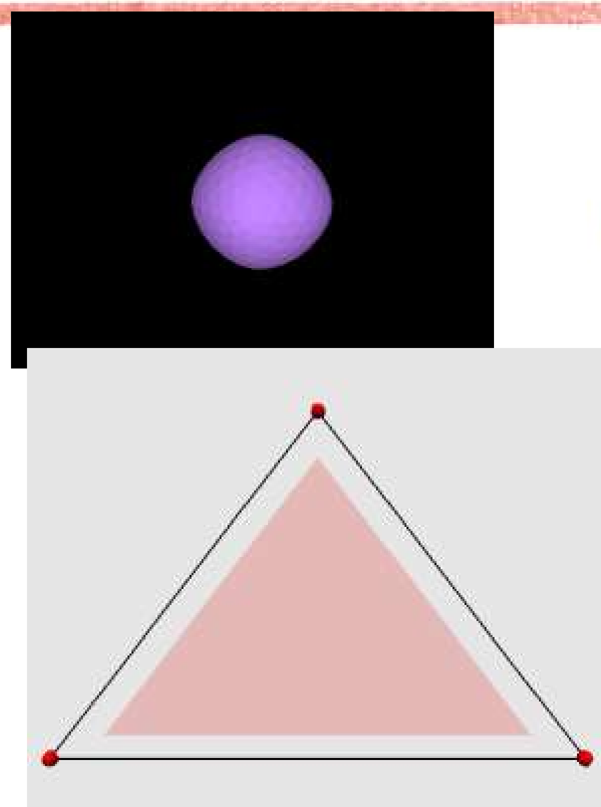
Patch: Mesh Subdivision



Triangular mesh

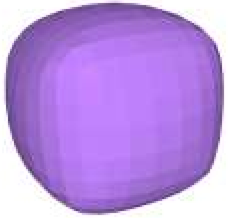


Quadrangular mesh



Butterfly

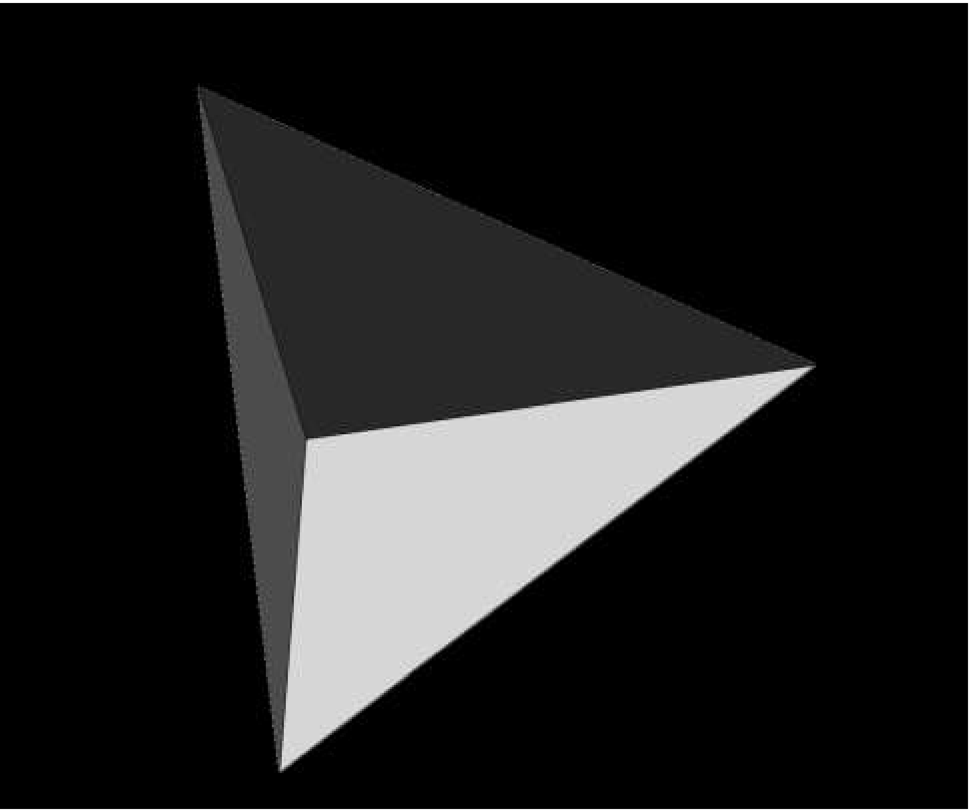
Kobbelt



Catmull-Clark

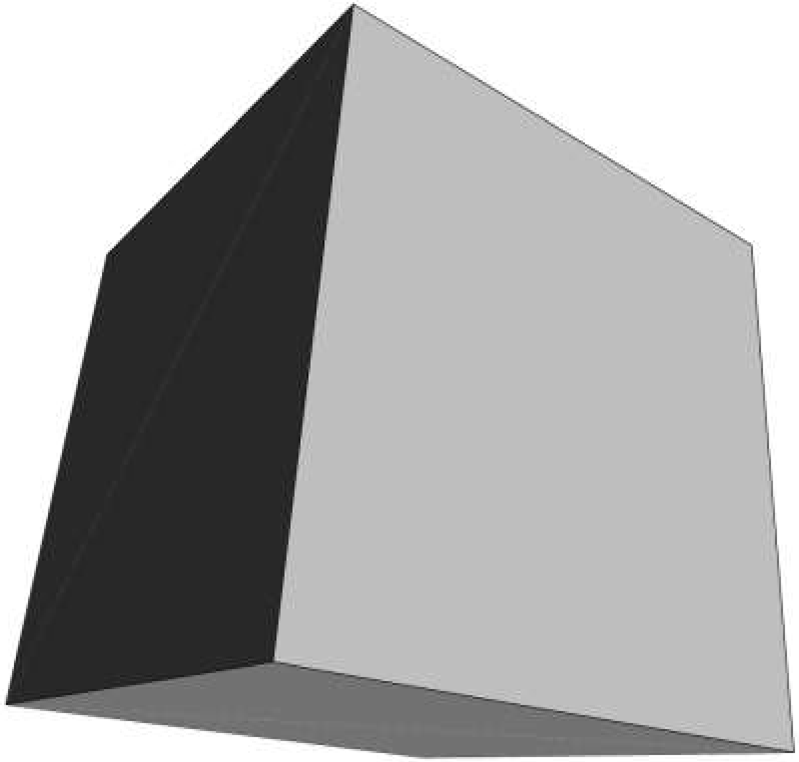
Doo-Sabin

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 $\mathcal{K} = (S, \prec)$
  - …
  - …
- **Topological Collections**
  - Formal Sums Representation $c \in C_S(V) \Longrightarrow c = \sum v_\sigma . \sigma$
  - 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 $r = \alpha \Longrightarrow \beta$
  - Topological Rewriting $c |\!\rhd_R c'$ where R is a set of rules

Diff. MGS: Algebraic Topology

Topological Chains

- Definition

> *Let $\mathcal{K}$ be an ACC and let $\boldsymbol{G}$ be an* ***abelian group****.*
> *A* **topological chain** *over $\mathcal{K}$ with values in $\boldsymbol{G}$ is a function* ***null almost everywhere*** *from $\mathcal{K}$ to $\boldsymbol{G}$. $\boldsymbol{C}_{\mathcal{K}}(\boldsymbol{G})$ denotes the topological chains over $\mathcal{K}$ with values in $\boldsymbol{G}$.*

- 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*

    $\boldsymbol{C}_{\mathcal{K}}(\boldsymbol{G})$ has an abelian group structure

Diff. MGS: Algebraic Topology

Topological **Co**chains

- Definition

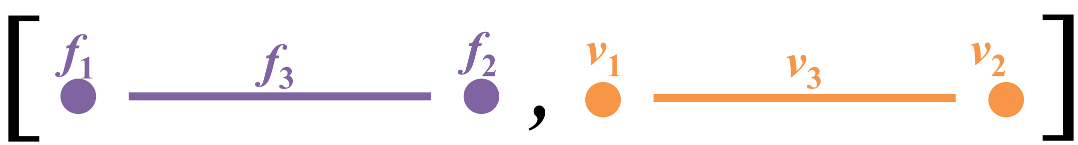
> *Let $\mathcal{K}$ be an ACC and let $\boldsymbol{G}$ and $\boldsymbol{H}$ be **abelian groups**.*
> *The **topological cochains** of chains of $\boldsymbol{C}_{\mathcal{K}}(\boldsymbol{G})$ into $\boldsymbol{H}$ are **group homomorphisms** from $\boldsymbol{C}_{\mathcal{K}}(\boldsymbol{G})$ to $\boldsymbol{H}$.*
> *$\boldsymbol{C}^{\mathcal{K}}(\boldsymbol{G},\boldsymbol{H})$ denotes the group of topological cochains from $\boldsymbol{C}_{\mathcal{K}}(\boldsymbol{G})$ to $\boldsymbol{H}$.*

- Representation with formal sums

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

- Application of a cochain on a chain

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



$$= f_1(v_1) + f_2(v_2) + f_3(v_3)$$

Diff. MGS: Summary

- Elements of Algebraic Topology
  - Abstract Cellular Complex $\mathcal{K} = (S, \prec)$
  - Topological Chain $c \in C_{\mathcal{K}}(G) \Longrightarrow c = \sum_{\sigma \in \mathcal{K}} v_\sigma . \sigma$
  - Topological Cochain $T \in C^{\mathcal{K}}(G, H) \Longrightarrow T = \sum_{\tau \in \mathcal{K}} f^\tau . \tau$
- Topological Collections
  - Formal Sums Representation $c \in C_S(V) \Longrightarrow c = \sum v_\sigma . \sigma$
  - Shape, Support $\mathrm{Shape}(c), |c|$
  - Sub-collection, Merge $s \subset c, c \uplus c'$
- Transformation
  - Collection Patterns/Expressions $\alpha \in C_{S^{\mathrm{var}}}(V^{\mathrm{var}}), \beta \in C_{S \cup S^{\mathrm{var}}}(\Sigma)$
  - Rewriting Rules $r = \alpha \Longrightarrow \beta$
  - Topological Rewriting $c | \rhd_R c'$ where R is a set of rules

Diff. MGS: Transformations vs. Cochains

- **Intersection Between Cochains and Transformations**
  - Topological Cochain $T \in C^{\mathcal{K}}(G,H) \Longrightarrow T = \sum_{\tau \in \mathcal{K}} f^{\tau}.\tau$
  - Topological Rewriting $c|\rhd_R c'$ where R is a set of rules
  - Rewriting Cochains
    - ***Cochains*** of $\boldsymbol{T} \in C^{\mathcal{K}}(G, C_{\mathcal{K}}(G)) = \mathrm{Hom}(C_{\mathcal{K}}(G), C_{\mathcal{K}}(G))$

      Mapping of topological chains to topological chains

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

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

      Application of a specific function on each cell of the collection

```
trans T = { x => f^x(x) }
```

    - One can show that $\boxed{\forall c \in C_{\mathcal{K}}(G) \quad c \;|\rhd_{\boldsymbol{R}} [\boldsymbol{T}, c]}$

Diff. MGS: Transport of Data

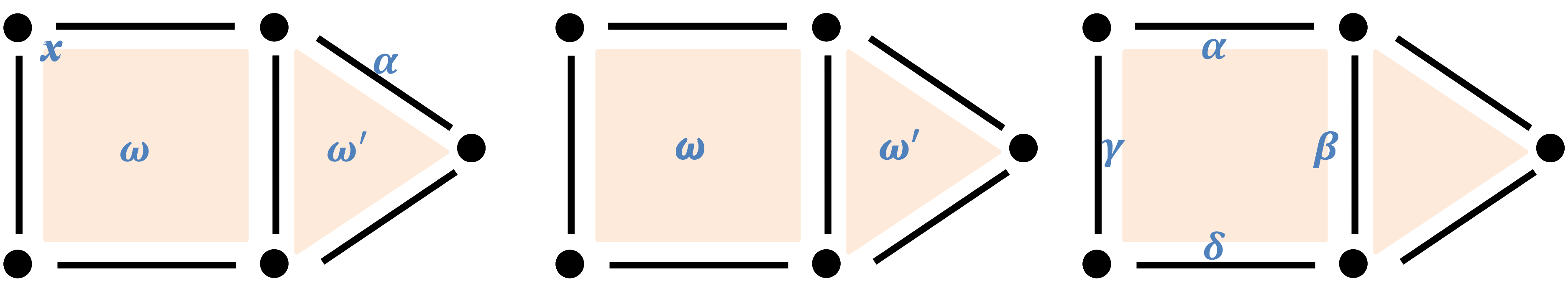
- Algebraic handling of collection
  - Usual functional map (when $f^x(X)$ does not depend on x)
  - Computing by moving data on the collection

    when $f^x(X)$ transports values from cells
    - to their p-neighbors (*i.e.*, the comma operator)

      trans *Eq1* = { x => **pNeighborsFold**(+, 0, x, ***p***) }
    - to their *faces* (*i.e.*, the face operator)

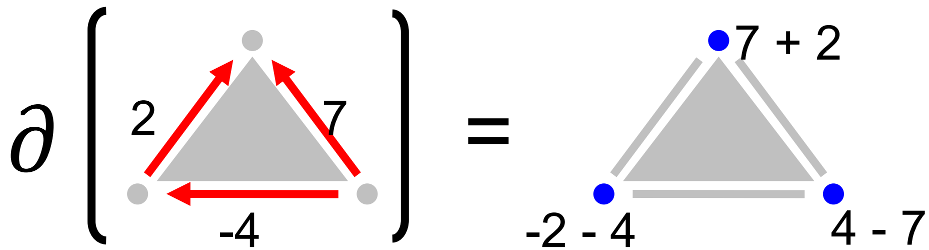
      trans *Eq2* = { x => **CofacesFold**(+, 0, x) }
    - to their *cofaces* (*i.e.*, the coface operator)

      trans *Eq3* = { x => **FacesFold**(+, 0, x) }



Differential Calculus in MGS

- The boundary operator ∂
  - Starting point of the elaboration of the *discrete differential calculus*



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

- The derivative operator $\mathbf{d}$
  - Defined w.r.t. *discrete Stockes' 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 Stockes' theorem

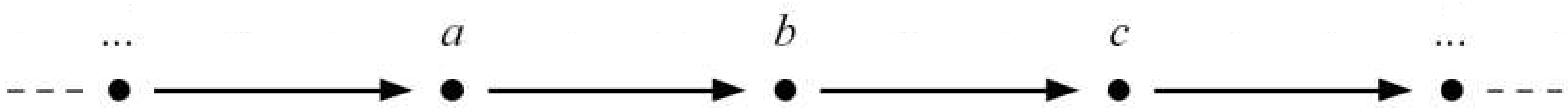
  - ***Coincides with Eq3*** (transport of data to cofaces) with orientation

Application: Generic Laplacian Operator

- Laplacian in Discrete Differential Calculus
  - Definition in terms of ∂ 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 ∂ 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(Derivative^co(T)))(c)
             + Sg(Derivative^co(Derivative(T)))(c)
```

Application: Generic Laplacian Operator

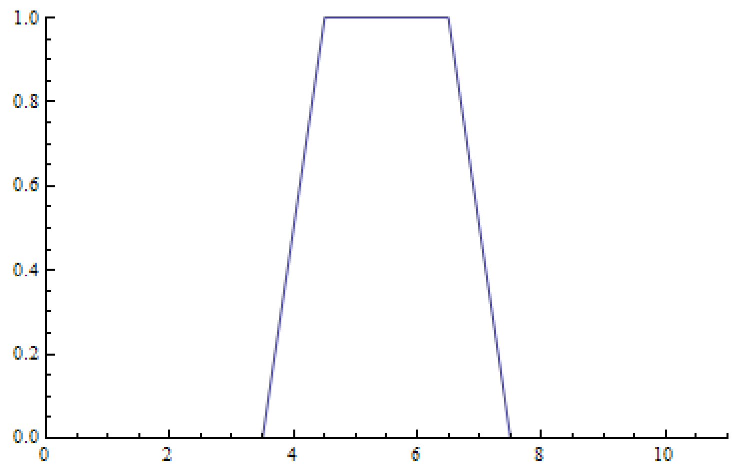
- **Generic Implantation of a Diffusion Operator**
  - Differential Equation & MGS implantation

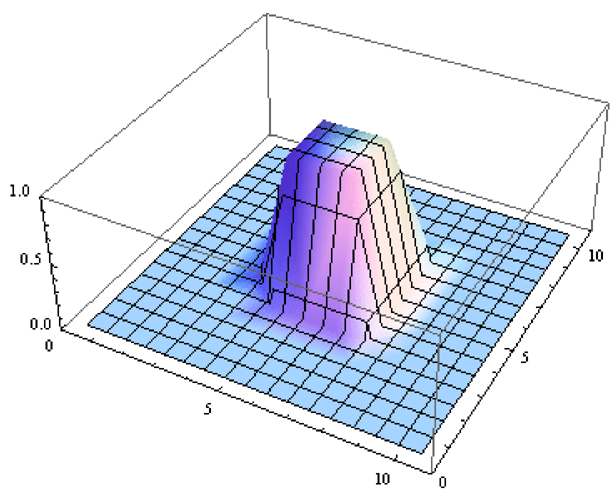
$$\frac{\partial u}{\partial t} = D \Delta u$$

```
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)

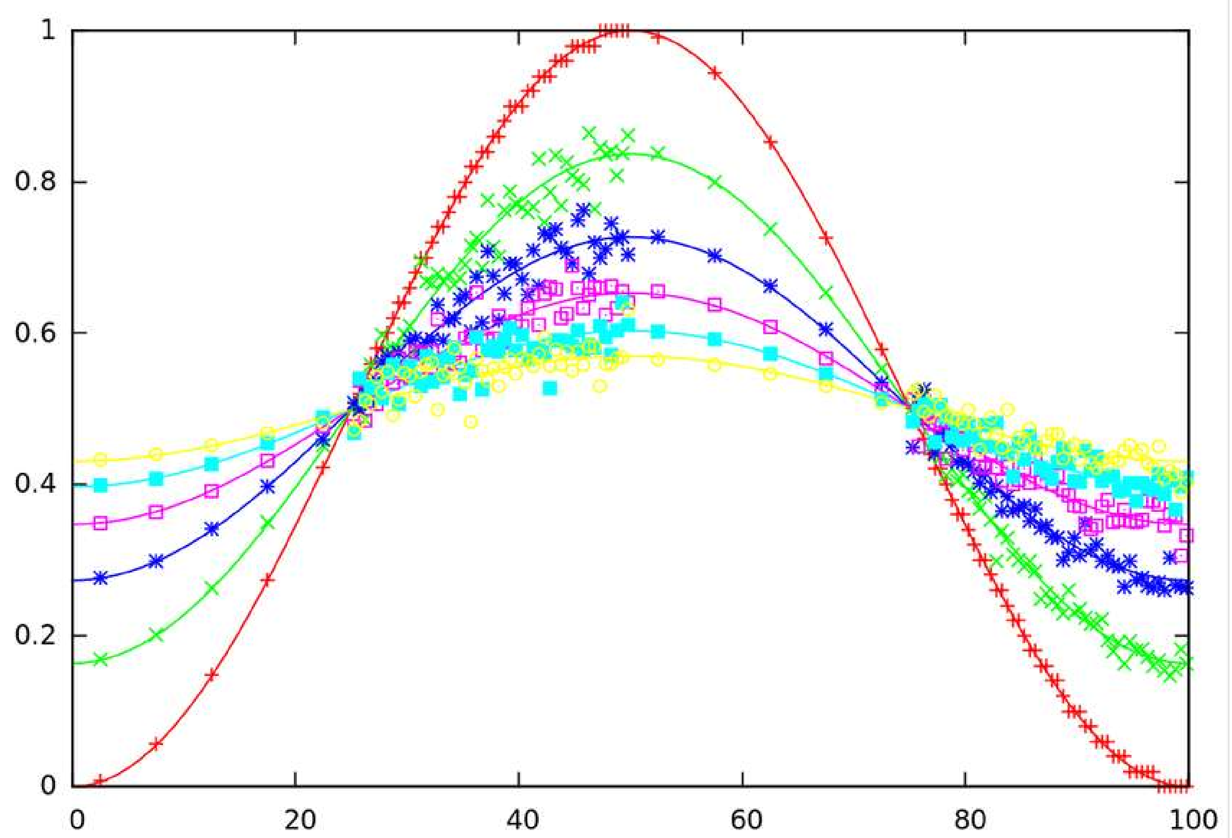




Application: Generic Laplacian Operator

- Generic Implantation of a Diffusion Operator
  - Stochastic Simulations

    Using another group G leads to random walk specifications



Application: Generic Laplacian Operator

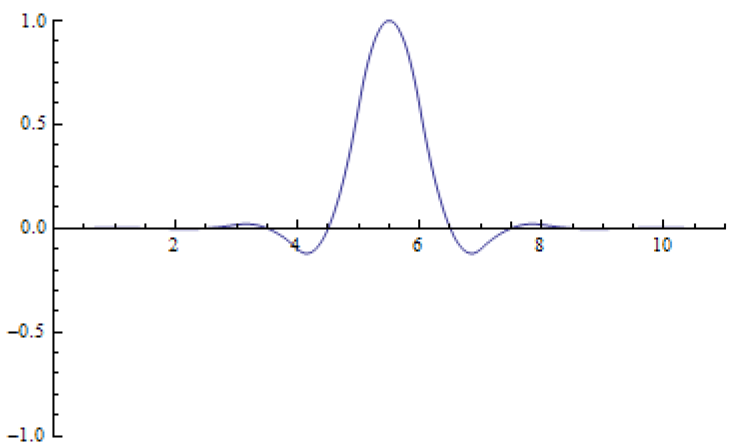
- Generic Implantation of a Wave Operator
  - Differential Equation & MGS implantation

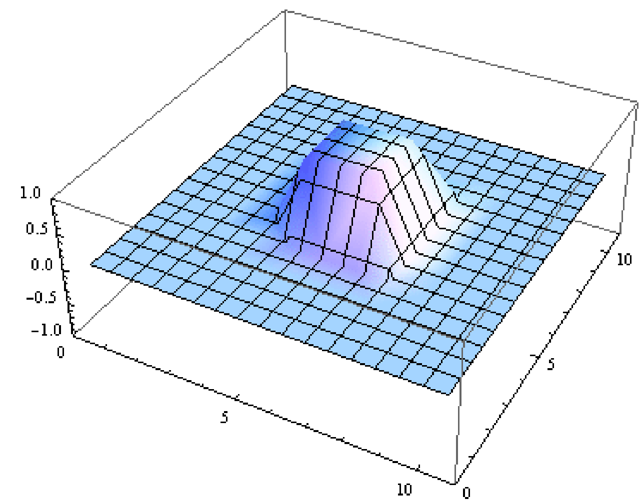
$$\frac{\partial^2 u}{\partial t^2} = C\Delta u$$

```
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)

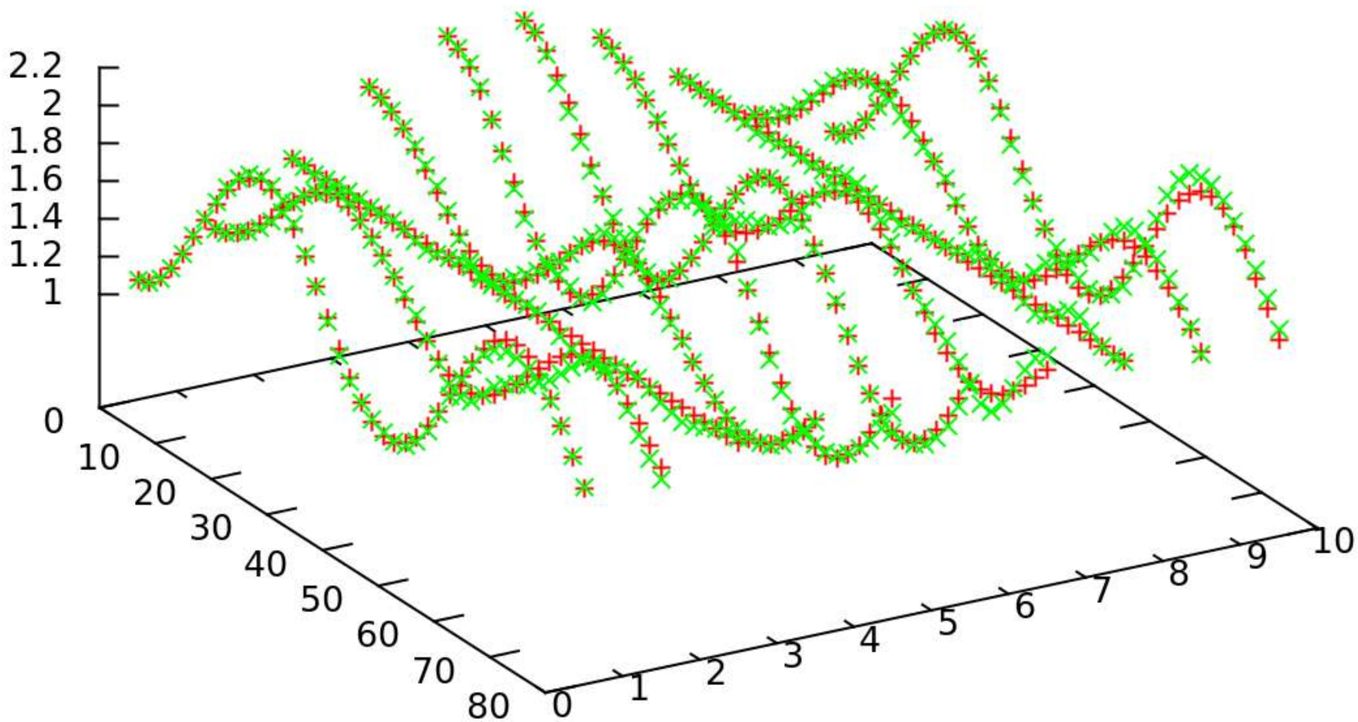




Application: Generic Laplacian Operator

- Generic Implantation of a Wave Operator
  - Stochastic Simulations

    Using another group G leads to random walk specifications

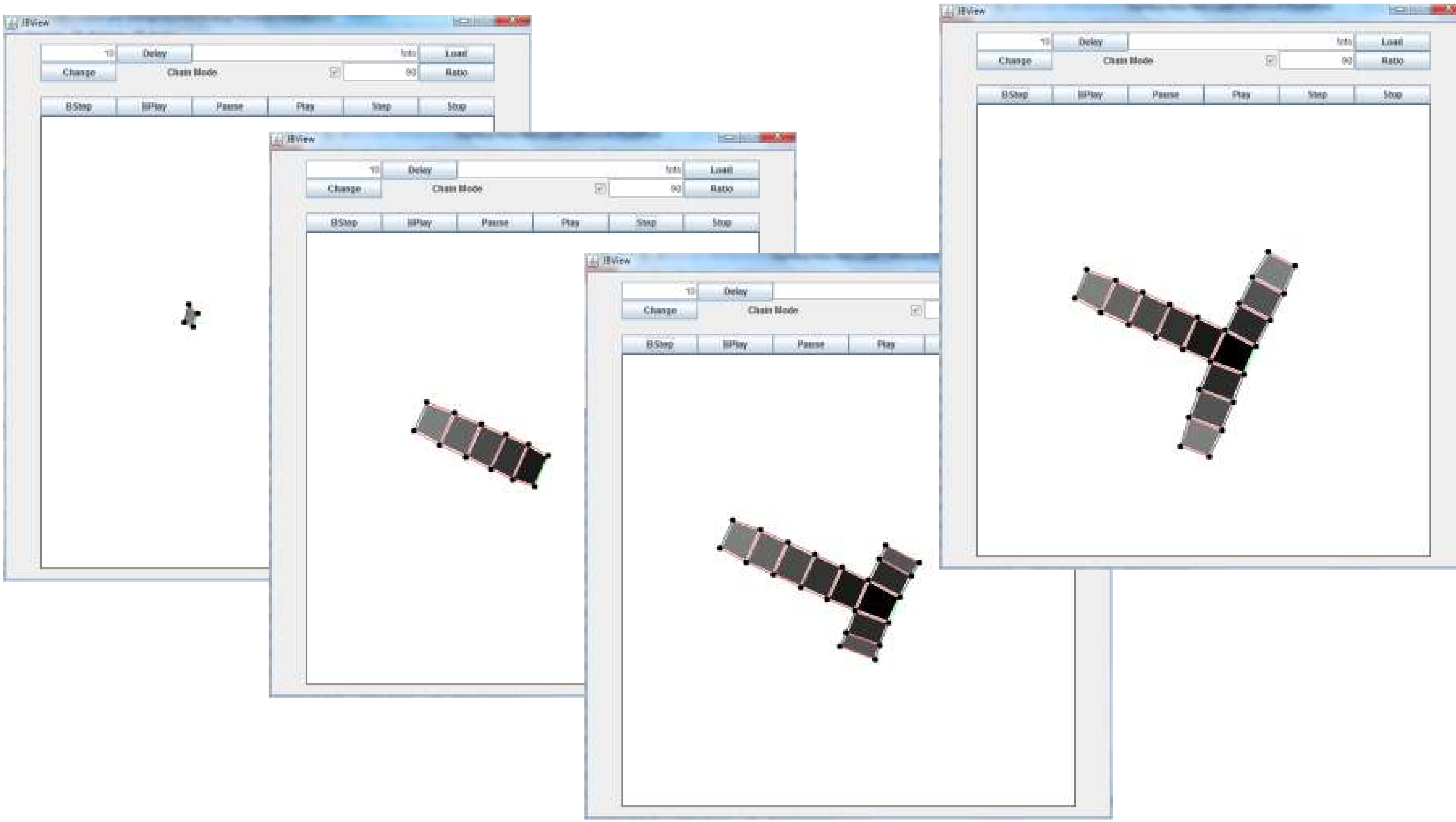


Outline

- 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

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

  - 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
}
```