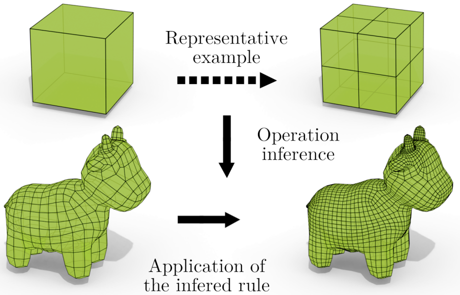
Graph transformation for reasoning about geometric modeling operations

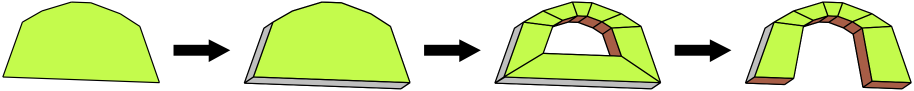


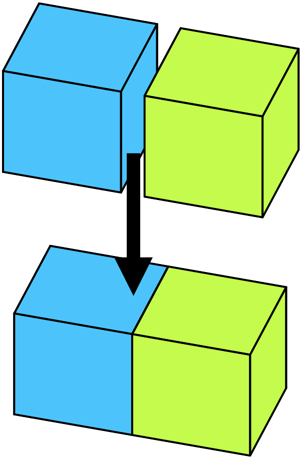
Romain Pascual

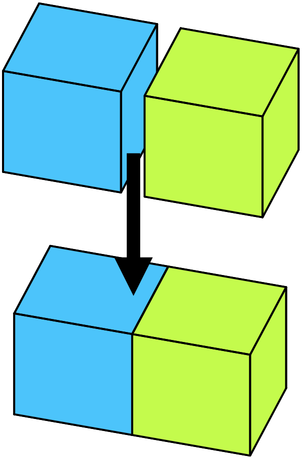
joint work with Pascale Le Gall, Hakim Belhaouari, and Agnès Arnould

March 30, 2023

MICS
université PARIS-SACLAY
xlim
Université de Poitiers

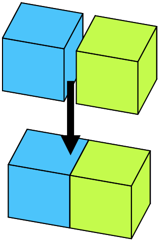






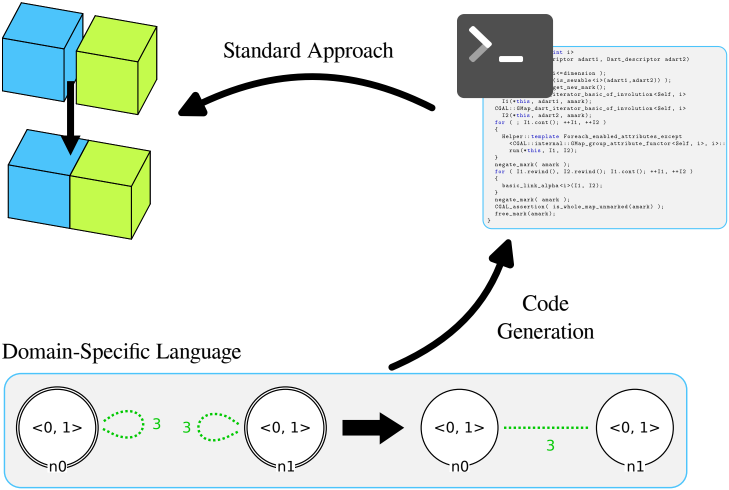
CGAL's sew operation

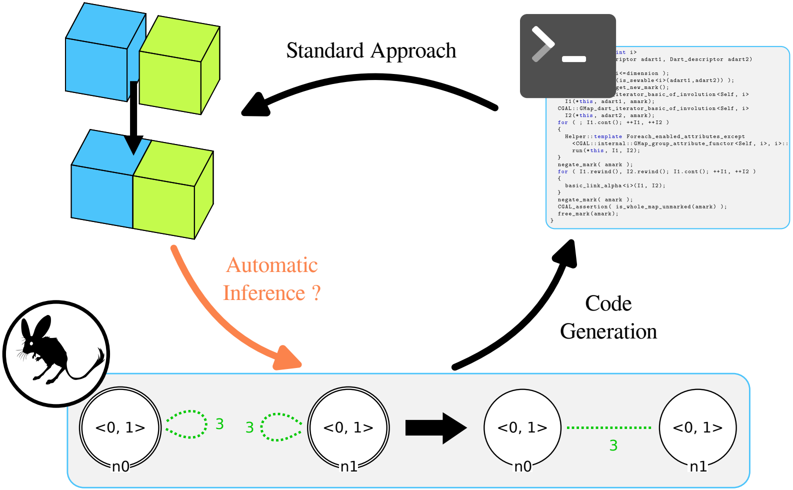
```
template<unsigned int i>
void sew(Dart_descriptor adart1, Dart_descriptor adart2)

  CGAL_assertion( i<=dimension );
  CGAL_assertion( (is_sewable<i>(adart1,adart2)) );
  size_type amark=get_new_mark();
  CGAL::GMap_dart_iterator_basic_of_involution<Self, i>
    I1(*this, adart1, amark);
  CGAL::GMap_dart_iterator_basic_of_involution<Self, i>
    I2(*this, adart2, amark);
  for ( ; I1.cont(); ++I1, ++I2 )
  {
    Helper::template Foreach_enabled_attributes_except
      <CGAL::internal::GMap_group_attribute_functor<Self, i>, i>::
      run(*this, I1, I2);
  }
  negate_mark( amark );
  for ( I1.rewind(), I2.rewind(); I1.cont(); ++I1, ++I2 )
  {
    basic_link_alpha<i>(I1, I2);
  }
  negate_mark( amark );
  CGAL_assertion( is_whole_map_unmarked(amark) );
  free_mark(amark);
}
```



Standard Approach

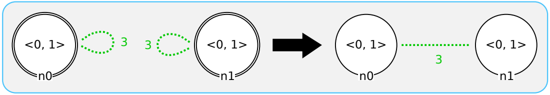
```
             int i>
             riptor adart1, Dart_descriptor adart2)

             i<=dimension );
             (is_sewable<i>(adart1,adart2)) );
             get_new_mark();
             iterator_basic_of_involution<Self, i>
    I1(*this, adart1, amark);
  CGAL::GMap_dart_iterator_basic_of_involution<Self, i>
    I2(*this, adart2, amark);
  for ( ; I1.cont(); ++I1, ++I2 )
  {
    Helper::template Foreach_enabled_attributes_except
      <CGAL::internal::GMap_group_attribute_functor<Self, i>, i>::
      run(*this, I1, I2);
  }
  negate_mark( amark );
  for ( I1.rewind(), I2.rewind(); I1.cont(); ++I1, ++I2 )
  {
    basic_link_alpha<i>(I1, I2);
  }
  negate_mark( amark );
  CGAL_assertion( is_whole_map_unmarked(amark) );
  free_mark(amark);
}
```


Standard Approach
Code Generation
Domain-Specific Language
<0, 1>
n0
3
3
<0, 1>
n1
<0, 1>
n0
3
<0, 1>
n1

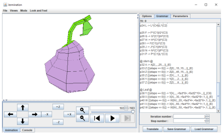

Standard Approach
Automatic
Inference ?
Code
Generation
int i>
riptor adart1, Dart_descriptor adart2)
i<=dimension);
(is_sewable<i>(adart1,adart2)));
get_new_mark();
iterator_basic_of_involution<Self, i>
I1(*this, adart1, amark);
CGAL::GMap_dart_iterator_basic_of_involution<Self, i>
I2(*this, adart2, amark);
for (; I1.cont(); ++I1, ++I2)
{
Helper::template Foreach_enabled_attributes_except
<CGAL::internal::GMap_group_attribute_functor<Self, i>, i>::
run(*this, I1, I2);
}
negate_mark(amark);
for (I1.rewind(), I2.rewind(); I1.cont(); ++I1, ++I2)
{
basic_link_alpha<i>(I1, I2);
}
negate_mark(amark);
CGAL_assertion(is_whole_map_unmarked(amark));
free_mark(amark);
}
<0, 1>
n0
3
3
<0, 1>
n1
<0, 1>
n0
3
<0, 1>
n1

Jerboa's DSL

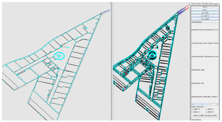
- Main characteristics:
  - Gmaps[1]
  - Syntax analyzer[2]



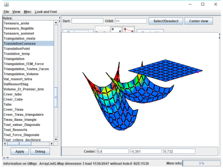
- Successful applications:
  - Plant growth



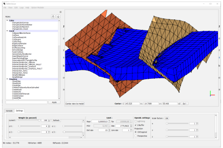
  - Architecture



  - Spring-mass

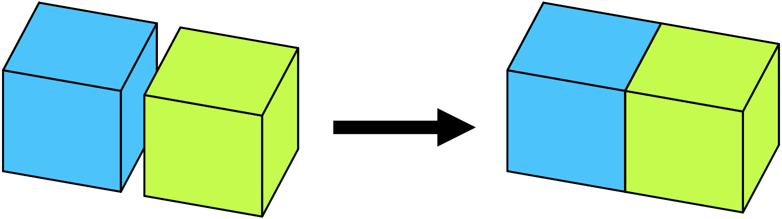


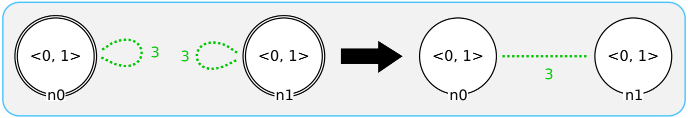
  - Geology



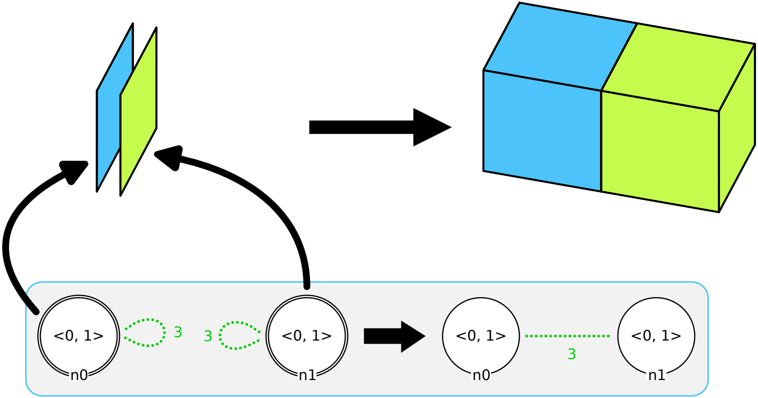
[1]Poudret et al. 2008
[2]Belhaouari et al. 2014

A sneak peek at Jerboa's language

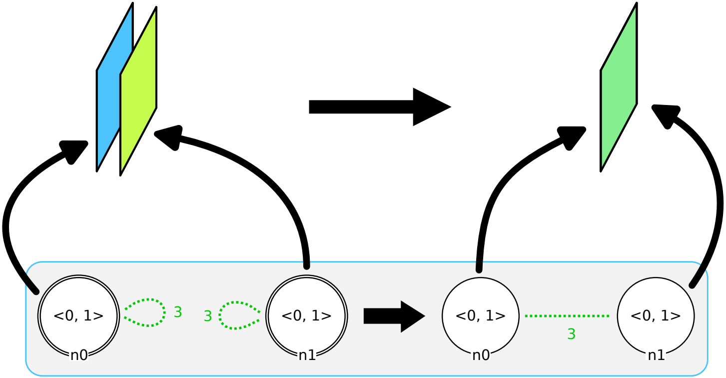




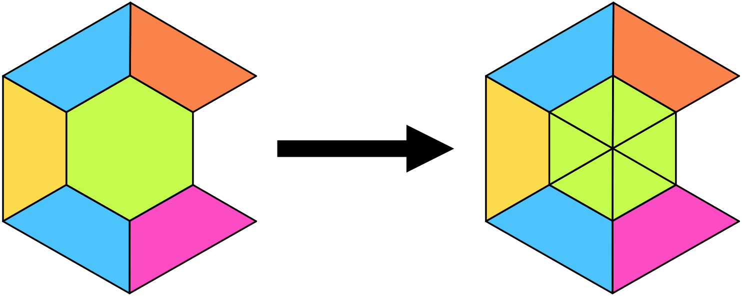
A sneak peek at Jerboa's language

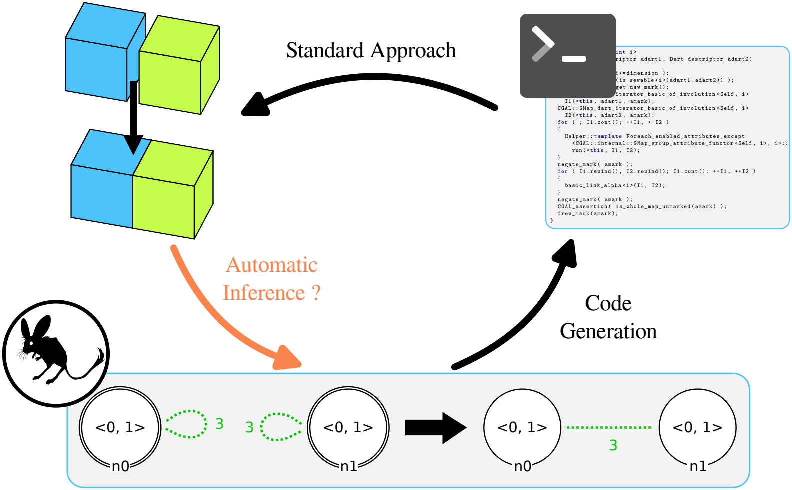


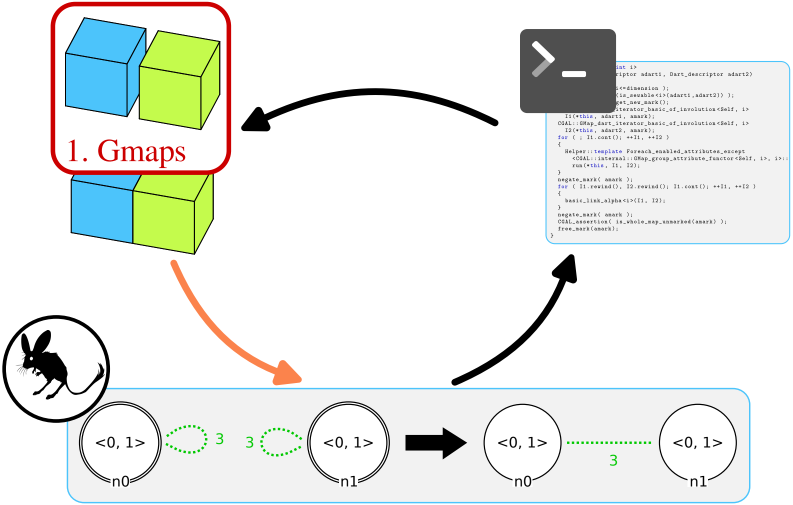
A sneak peek at Jerboa's language

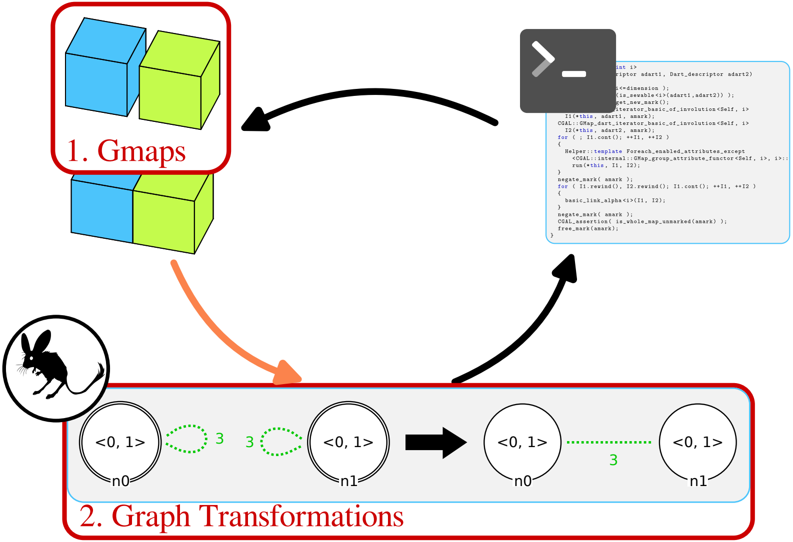


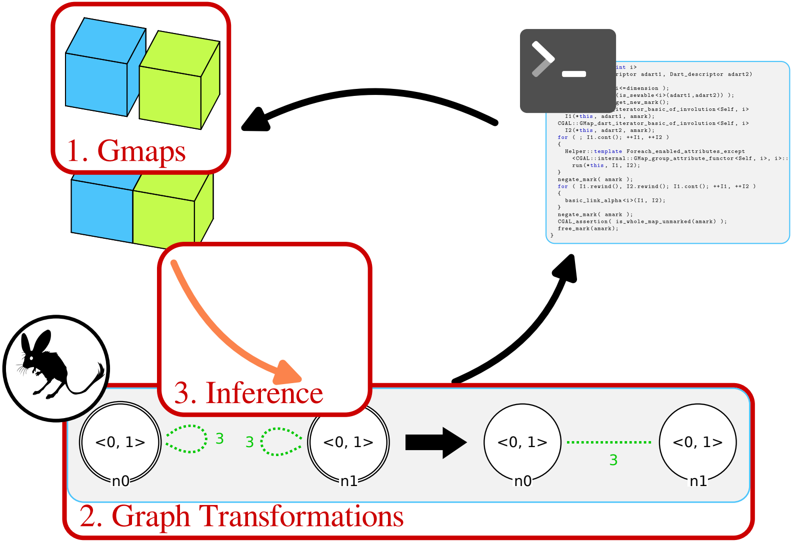
Running example: face triangulation

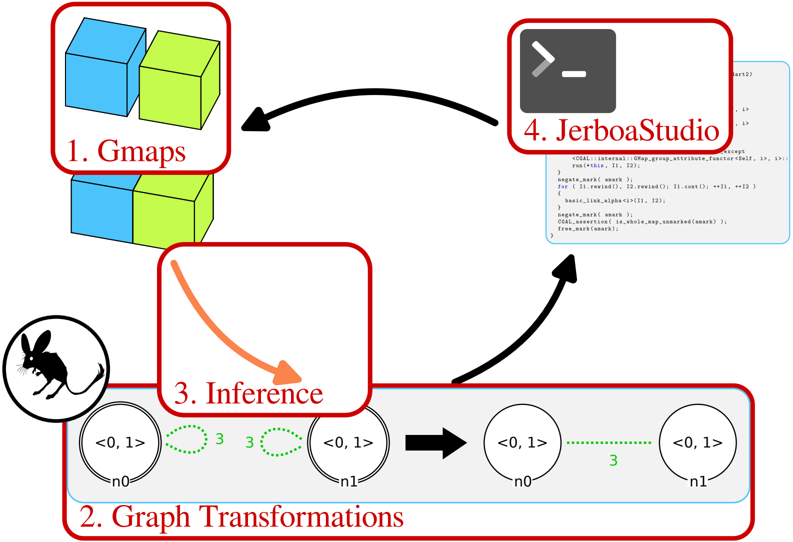



Standard Approach
Automatic
Inference ?
Code
Generation
I1(*this, adart1, amark);
CGAL::GMap_dart_iterator_basic_of_involution<Self, i>
I2(*this, adart2, amark);
for (; I1.cont(); ++I1, ++I2)
{
Helper::template Foreach_enabled_attributes_except
<CGAL::internal::GMap_group_attribute_functor<Self, i>, i>::
run(*this, I1, I2);
}
negate_mark(amark);
for (I1.rewind(), I2.rewind(); I1.cont(); ++I1, ++I2)
{
basic_link_alpha<i>(I1, I2);
}
negate_mark(amark);
CGAL_assertion(is_whole_map_unmarked(amark));
free_mark(amark);
}
<0, 1>
n0
3
3
<0, 1>
n1
<0, 1>
n0
3
<0, 1>
n1


1. Gmaps
int i>
riptor adart1, Dart_descriptor adart2)
i<=dimension);
(is_sewable<i>(adart1,adart2)));
get_new_mark();
iterator_basic_of_involution<Self, i>
I1(*this, adart1, amark);
CGAL::GMap_dart_iterator_basic_of_involution<Self, i>
I2(*this, adart2, amark);
for (; I1.cont(); ++I1, ++I2)
{
Helper::template Foreach_enabled_attributes_except
<CGAL::internal::GMap_group_attribute_functor<Self, i>, i>::
run(*this, I1, I2);
}
negate_mark(amark);
for (I1.rewind(), I2.rewind(); I1.cont(); ++I1, ++I2)
{
basic_link_alpha<i>(I1, I2);
}
negate_mark(amark);
CGAL_assertion(is_whole_map_unmarked(amark));
free_mark(amark);
}
<0, 1>
n0
3
3
<0, 1>
n1
<0, 1>
n0
3
<0, 1>
n1


1. Gmaps
int i>
riptor adart1, Dart_descriptor adart2)
i<=dimension);
(is_sewable<i>(adart1,adart2)));
get_new_mark();
iterator_basic_of_involution<Self, i>
I1(*this, adart1, amark);
CGAL::GMap_dart_iterator_basic_of_involution<Self, i>
I2(*this, adart2, amark);
for (; I1.cont(); ++I1, ++I2)
{
Helper::template Foreach_enabled_attributes_except
<CGAL::internal::GMap_group_attribute_functor<Self, i>, i>::
run(*this, I1, I2);
}
negate_mark(amark);
for (I1.rewind(), I2.rewind(); I1.cont(); ++I1, ++I2)
{
basic_link_alpha<i>(I1, I2);
}
negate_mark(amark);
CGAL_assertion(is_whole_map_unmarked(amark));
free_mark(amark);
}
<0, 1>
n0
3
3
<0, 1>
n1
<0, 1>
n0
3
<0, 1>
n1
2. Graph Transformations


1. Gmaps
3. Inference
2. Graph Transformations
<0, 1>
n0
3
3
<0, 1>
n1
<0, 1>
n0
3
<0, 1>
n1


1. Gmaps
4. JerboaStudio
3. Inference
2. Graph Transformations
<0, 1>
n0
3
3
<0, 1>
n1
<0, 1>
n0
3
<0, 1>
n1

Generalized maps

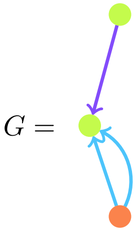
- Geometric objects are represented with embedded generalized maps.

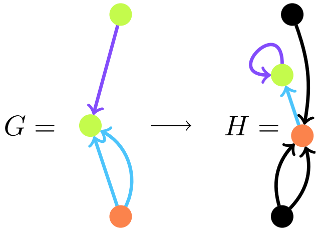
The category of graphs

$G =$ 

A graph $G = (V, E, s, t)$:

- a set of nodes V,
- a set of arcs E,
- a source function $s : E \to V$,
- a target arrow $t : E \to V$,





A graph $G = (V, E, s, t)$:

- a set of nodes V,
- a set of arcs E,
- a source function $s : E \to V$,
- a target arrow $t : E \to V$,

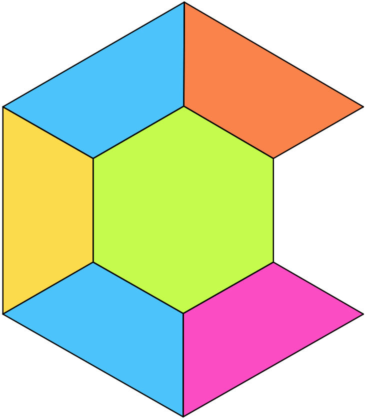
A morphism $G \to H$:

- a node function $V_G \to V_H$,
- an arc function $E_G \to E_H$,

preserving structure.

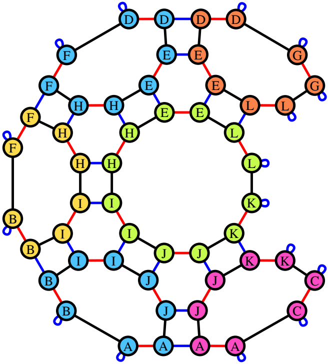
- Decorated with labels, types, and attributes.

Generalized maps[1]



[1]Damiand et al. 2014.

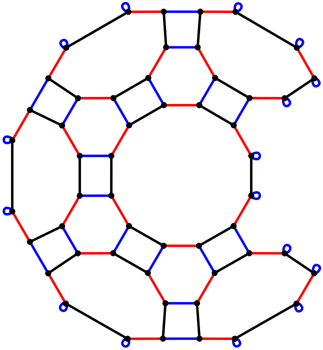
Generalized maps[1]



Color legend: 0, 1, 2.

[1] Damiand et al. 2014.

Generalized maps[1]

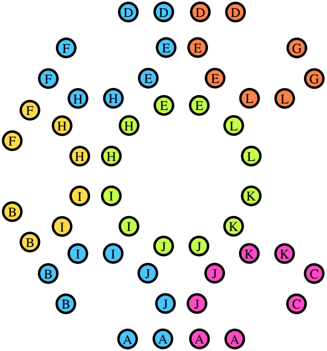


Color legend: 0, 1, 2.

- Topology: graph structure

[1] Damiand et al. 2014.

Generalized maps[1]

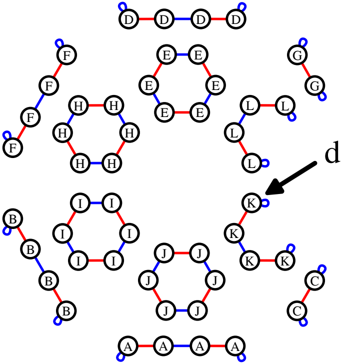


- Topology: graph structure
- Geometry: node attributes

Color legend: 0, 1, 2.

[1] Damiand et al. 2014.

Orbits and topological cells

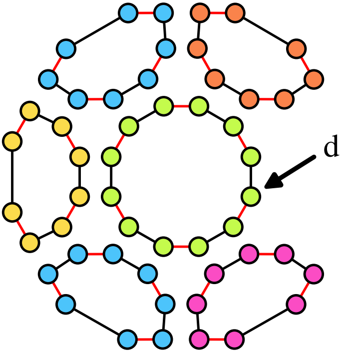


Color legend: 0, 1, 2.

Orbit (encode topological cell):
Graph induced by a subset $\langle o \rangle \subseteq [\![0, n]\!]$ of dimensions.

- positions on vertices (orbits $\langle 1, 2 \rangle$).

Orbits and topological cells



Color legend: 0, 1, 2.

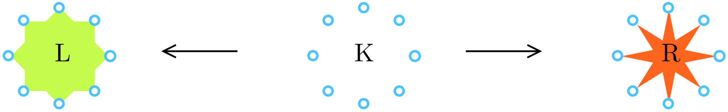
Orbit (encode topological cell):
Graph induced by a subset $\langle o \rangle \subseteq [\![0, n]\!]$ of dimensions.

- positions on vertices (orbits $\langle 1, 2 \rangle$).
- colors on faces (orbits $\langle 0, 1 \rangle$).

Graph rewriting

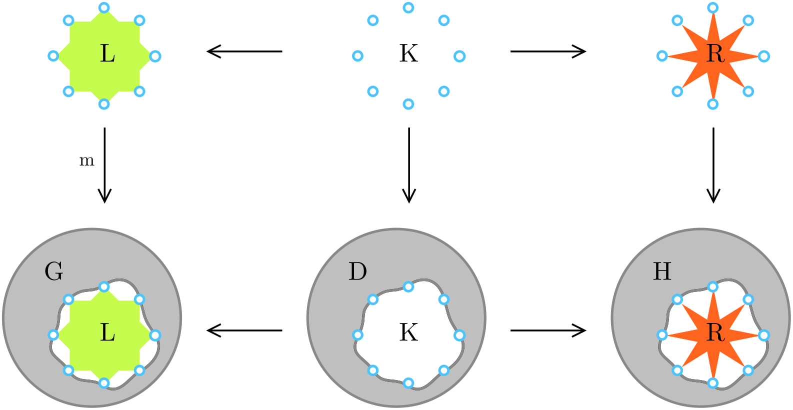
- Operations on Gmaps are designed as graph rewriting rules.

Graph transformation rules[1]



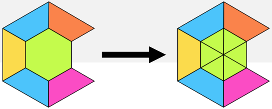
[1] Rozenberg 1997; Ehrig et al. 2006; Heckel et al. 2020.

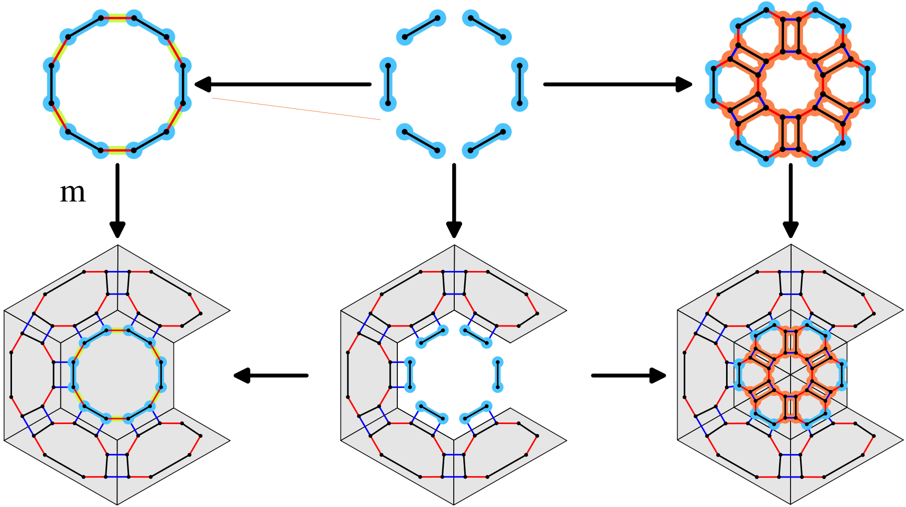
Graph transformation rules[1]



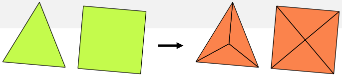
[1] Rozenberg 1997; Ehrig et al. 2006; Heckel et al. 2020.

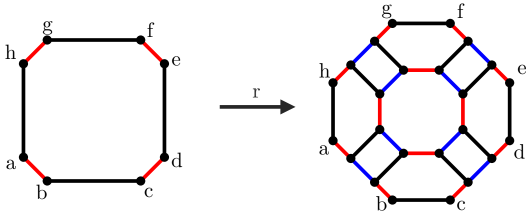
Rewriting Gmaps



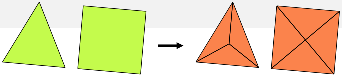


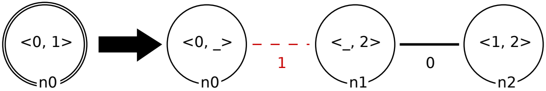
Orbit rewriting

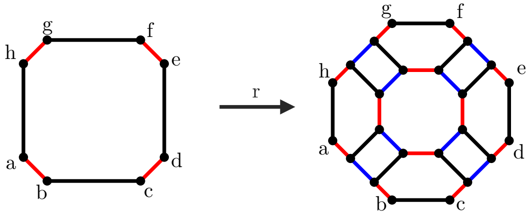




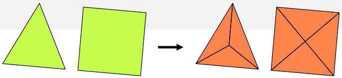
Orbit rewriting

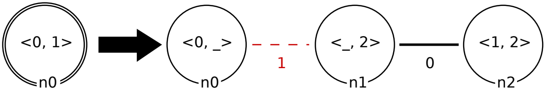


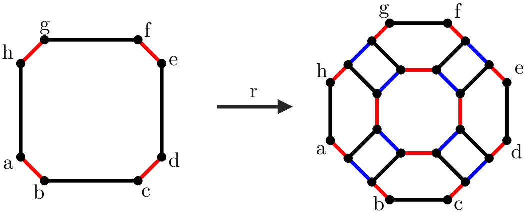




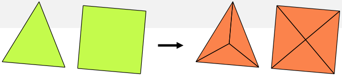
Orbit rewriting

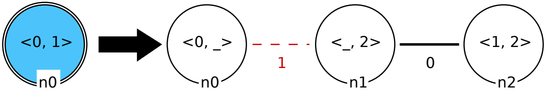


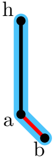


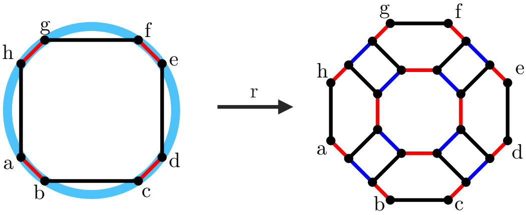


Orbit rewriting

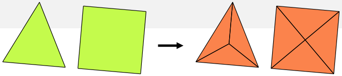


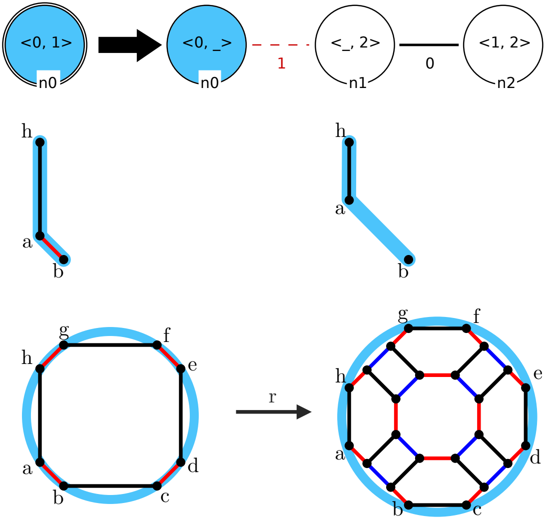




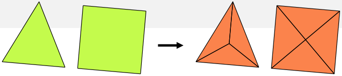


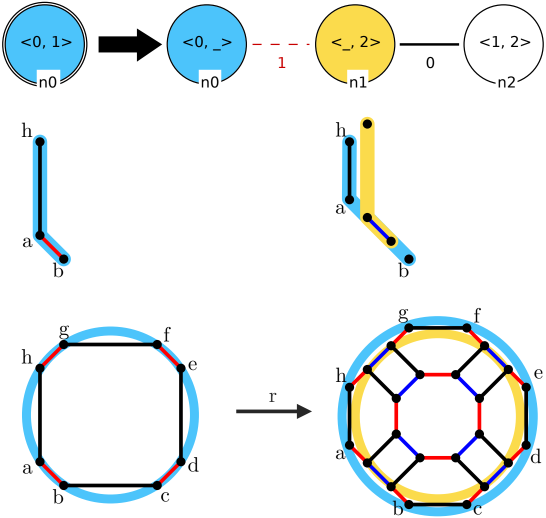
Orbit rewriting



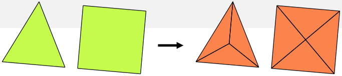


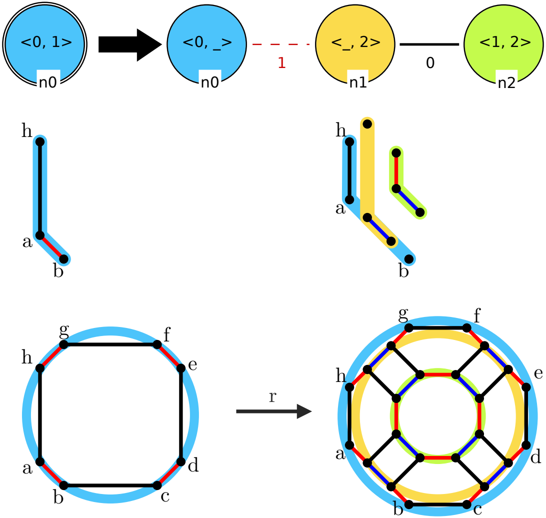
Orbit rewriting



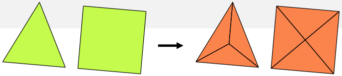


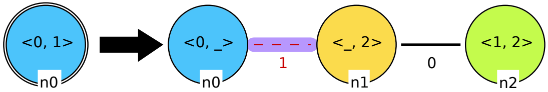
Orbit rewriting

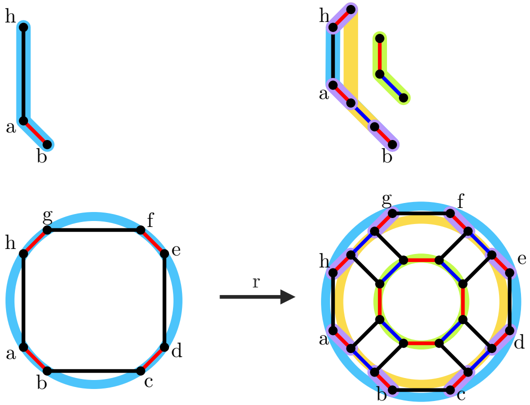




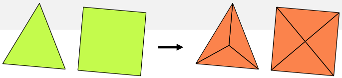
Orbit rewriting

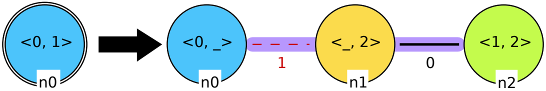


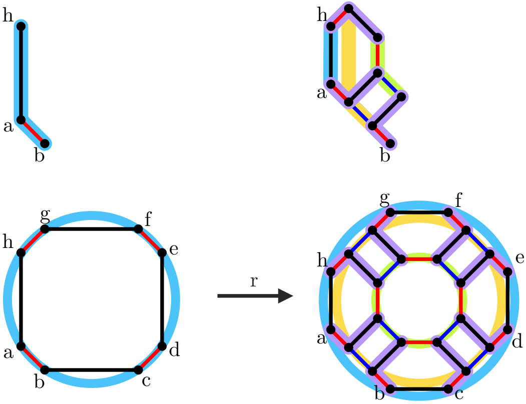




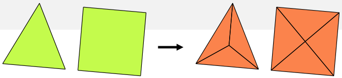
Orbit rewriting

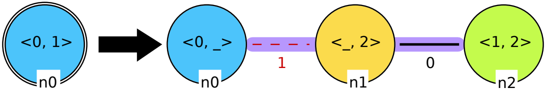


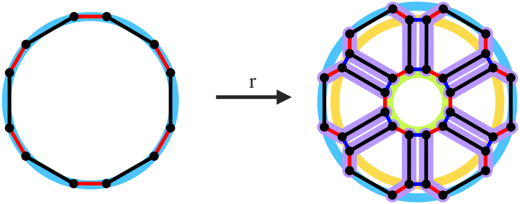




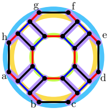
Orbit rewriting

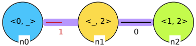


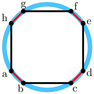




Graph products[1]

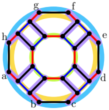


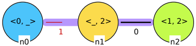


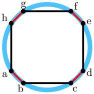


[1] inspired from Bauderon 1995.

Graph products[1]



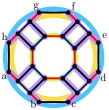


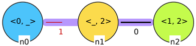


P

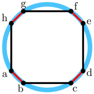
[1] inspired from Bauderon 1995.

Graph products[1]





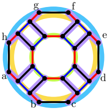
Π



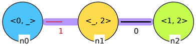
P

[1]inspired from Bauderon 1995.

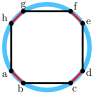
Graph products[1]



$\iota(\Pi, P)$



Π

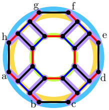


P

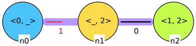
- $\iota(\Pi, P)$: instantiation.

[1] inspired from Bauderon 1995.

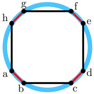
Graph products[1]



$\iota(\Pi, P)$



Π

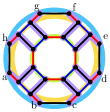


$$P \xRightarrow{\mathbb{E}_\Sigma} \mathbb{E}_\Sigma(P)$$

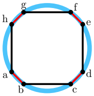
- $\iota(\Pi, P)$: instantiation.
- $\mathbb{E}_\Sigma$: embedding functor.

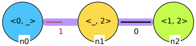
[1] inspired from Bauderon 1995.

Graph products[1]



$\iota(\Pi, P)$

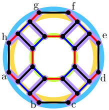


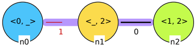


$$\begin{tikzcd} & \mathbb{E}_\Sigma(P) \times \Pi \arrow[r] \arrow[d] \arrow[dr, phantom, "PB"] & \Pi \arrow[d, "!_\Pi"] \\ P \arrow[r, Rightarrow, "\mathbb{E}_\Sigma"] & \mathbb{E}_\Sigma(P) \arrow[r, "!_{\mathbb{E}_\Sigma(P)}"] & 1_{\Sigma^2} \end{tikzcd}$$

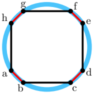
- $\iota(\Pi, P)$: instantiation.
- $\mathbb{E}_\Sigma$: embedding functor.

[1] inspired from Bauderon 1995.





$$
\begin{tikzcd}
\iota(\Pi, P) & \mathbb{E}_\Sigma(P) \times \Pi \arrow[l, Rightarrow, "\pi_\Sigma"'] \arrow[r] \arrow[d] \arrow[dr, phantom, "PB"] & \Pi \arrow[d, "!_\Pi"] \\
P \arrow[r, Rightarrow, "\mathbb{E}_\Sigma"] & \mathbb{E}_\Sigma(P) \arrow[r, "!_{\mathbb{E}_\Sigma(P)}"] & 1_{\Sigma^2}
\end{tikzcd}
$$



- $\iota(\Pi, P)$: instantiation.
- $\mathbb{E}_\Sigma$: embedding functor.
- π_Σ: projecting functor.

[1] inspired from Bauderon 1995.

Application of a rule scheme

$$
\begin{tikzcd}
1_{\mathbb{W}\times\mathbb{D}} & & & \\
E_{\mathbb{D}}(P_v) \arrow[u, "!_{E_{\mathbb{D}}(P_v)}"] & L \arrow[ul, "!_L"] & K \arrow[l, hook'] \arrow[r, hook] \arrow[ull, "!_K"] & R \arrow[ulll, "!_R"] \\
P_v \arrow[u, Rightarrow, "E_{\mathbb{D}}"] \arrow[d, hook, "p_v"'] & E_{\mathbb{D}}(P_v)\times L \arrow[u] \arrow[ul] \arrow[d, Rightarrow, "\pi_{\mathbb{D}}"] & E_{\mathbb{D}}(P_v)\times K \arrow[l, hook'] \arrow[r, hook] \arrow[u] \arrow[ull] \arrow[d, Rightarrow, "\pi_{\mathbb{D}}"] & E_{\mathbb{D}}(P_v)\times R \arrow[u] \arrow[ulll] \arrow[d, Rightarrow, "\pi_{\mathbb{D}}"] \\
\mathbb{P}(\mathcal{G}) & \iota(L,P_v) \arrow[d, hook, "m"] & \iota(K,P_v) \arrow[l, hook'] \arrow[r, hook] \arrow[d, hook] & \iota(R,P_v) \arrow[d, hook] \\
 & \mathcal{G} \arrow[ul, Rightarrow, "\mathbb{P}"] & \mathcal{D} \arrow[l, hook'] \arrow[r, hook] & \mathcal{H}
\end{tikzcd}
$$

Application of a rule scheme

$$
\begin{tikzcd}
1_{\mathbb{W}\times\mathbb{D}} & & & \\
E_{\mathbb{D}}(P_v) \arrow[u, "!_{E_{\mathbb{D}}(P_v)}"] & L \arrow[ul, two heads, "!_L"'] & K \arrow[l, hook'] \arrow[r, hook] \arrow[ull, two heads, "!_K"'] & R \arrow[ulll, two heads, "!_R"'] \\
& \text{Input 1: Rule scheme} & & \\
P_v \arrow[uu, Rightarrow, "E_{\mathbb{D}}"] \arrow[d, hook, "p_v"'] & E_{\mathbb{D}}(P_v)\times L \arrow[uul, two heads] \arrow[uu, dash] \arrow[d, Rightarrow, "\pi_{\mathbb{D}}"] & E_{\mathbb{D}}(P_v)\times K \arrow[l, hook'] \arrow[r, hook] \arrow[uull, two heads] \arrow[uu, dash] \arrow[d, Rightarrow, "\pi_{\mathbb{D}}"] & E_{\mathbb{D}}(P_v)\times R \arrow[uulll, two heads] \arrow[uu, dash] \arrow[d, Rightarrow, "\pi_{\mathbb{D}}"] \\
\mathbb{P}(\mathcal{G}) & \iota(L,P_v) \arrow[d, hook, "m"] & \iota(K,P_v) \arrow[l, hook'] \arrow[r, hook] \arrow[d, hook] & \iota(R,P_v) \arrow[d, hook] \\
& \begin{array}{l}\mathcal{G}\\ \text{Input 2: Gmap}\end{array} \arrow[ul, Rightarrow, "\mathbb{P}"] & \mathcal{D} \arrow[l, hook'] \arrow[r, hook] & \begin{array}{l}\mathcal{H}\\ \text{Output: Gmap}\end{array}
\end{tikzcd}
$$

Application of a rule scheme

$$
\begin{tikzcd}
1_{\mathbb{W}\times\mathbb{D}} & & & \\
E_{\mathbb{D}}(P_v) \arrow[u, "!_{E_{\mathbb{D}}(P_v)}"] & L \arrow[ul, "!_L"'] & K \arrow[l, hook'] \arrow[r, hook] \arrow[ull, "!_K"'] & R \arrow[ulll, "!_R"'] \\
P_v \arrow[u, Rightarrow, "E_{\mathbb{D}}"] \arrow[d, "p_v"'] & E_{\mathbb{D}}(P_v)\times L \arrow[u] \arrow[ul] \arrow[d, Rightarrow, "\pi_{\mathbb{D}}"] & E_{\mathbb{D}}(P_v)\times K \arrow[u] \arrow[l, hook'] \arrow[r, hook] \arrow[ull] \arrow[d, Rightarrow, "\pi_{\mathbb{D}}"] & E_{\mathbb{D}}(P_v)\times R \arrow[u] \arrow[ulll] \arrow[d, Rightarrow, "\pi_{\mathbb{D}}"] \\
\mathbb{P}(\mathcal{G}) & \iota(L,P_v) \arrow[d] & \iota(K,P_v) \arrow[l, hook'] \arrow[r, hook] \arrow[d, hook] & \iota(R,P_v) \arrow[d, hook] \\
 & \mathcal{G} \arrow[ul, Rightarrow, "\mathbb{P}"] & \mathcal{D} \arrow[l, hook'] \arrow[r, hook] & \mathcal{H}
\end{tikzcd}
$$

Product

Application of a rule scheme

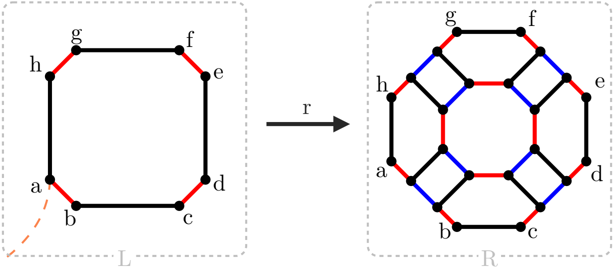
$$
\begin{tikzcd}
1_{\mathbb{W}\times\mathbb{D}} & & & \\
E_{\mathbb{D}}(P_v) \arrow[u, "!_{E_{\mathbb{D}}(P_v)}"] & L \arrow[ul, "!_L"'] & K \arrow[l] \arrow[r, hook] \arrow[ull, "!_K"'] & R \arrow[ulll, "!_R"'] \\
P_v \arrow[u, Rightarrow, "E_{\mathbb{D}}"] \arrow[d, hook, "p_v"'] & E_{\mathbb{D}}(P_v)\times L \arrow[u] \arrow[ul] \arrow[d, Rightarrow, "\pi_{\mathbb{D}}"] & E_{\mathbb{D}}(P_v)\times K \arrow[l, hook'] \arrow[r, hook] \arrow[u] \arrow[ull] \arrow[d, Rightarrow, "\pi_{\mathbb{D}}"] & E_{\mathbb{D}}(P_v)\times R \arrow[u] \arrow[ulll] \arrow[d, Rightarrow, "\pi_{\mathbb{D}}"] \\
\mathbb{P}(\mathcal{G}) & \iota(L,P_v) \arrow[d, hook, "m"] & \iota(K,P_v) \arrow[l, hook'] \arrow[r, hook] \arrow[d, hook] & \iota(R,P_v) \arrow[d, hook] \\
 & \mathcal{G} \arrow[ul, Rightarrow, "\mathbb{P}"] & \mathcal{D} \arrow[l, hook'] \arrow[r, hook] & \mathcal{H}
\end{tikzcd}
$$

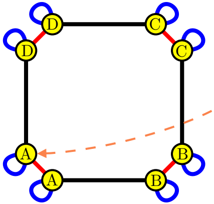
'Standard' DPO step

Modifying geometric values[1]

[1] Bellet et al. 2017.

Modifying geometric values[1]



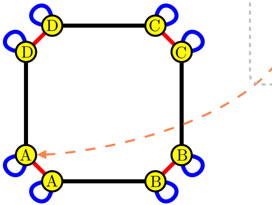


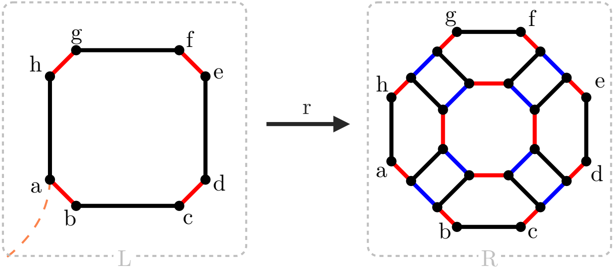
Algebraic data types:

- add
- middle
- scale
- ...

[1]Bellet et al. 2017.

Modifying geometric values[1]



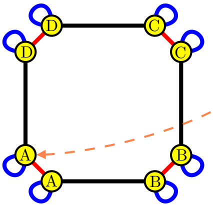


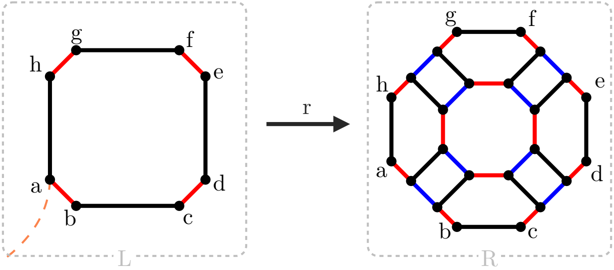
Extended with topological operators:

- Neighbor operator:
  - $a@0@1@0.position = f.position = C$
  - $a@1@0.color = c.color =$ ●

[1] Bellet et al. 2017.

Modifying geometric values[1]



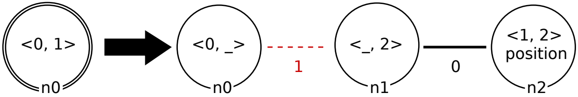


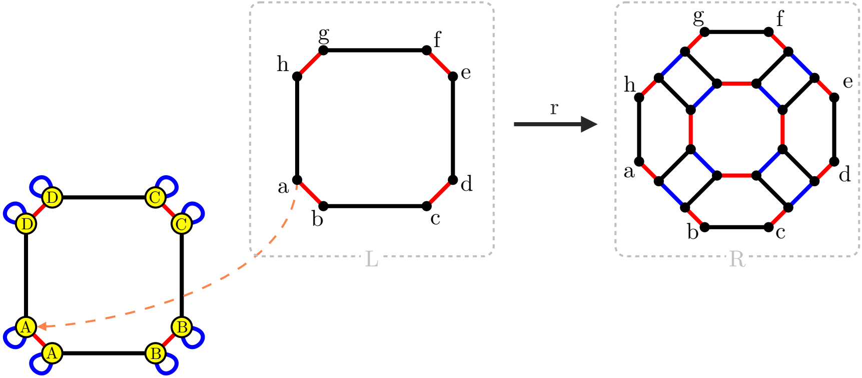
Extended with topological operators:

- Neighbor operator:
- Collect operator:
  - $position_{\langle 0,1 \rangle}(a) = \{A, B, C, D\}$
  - $color_{\langle 0,1 \rangle}(a) = \{ \bullet \}$

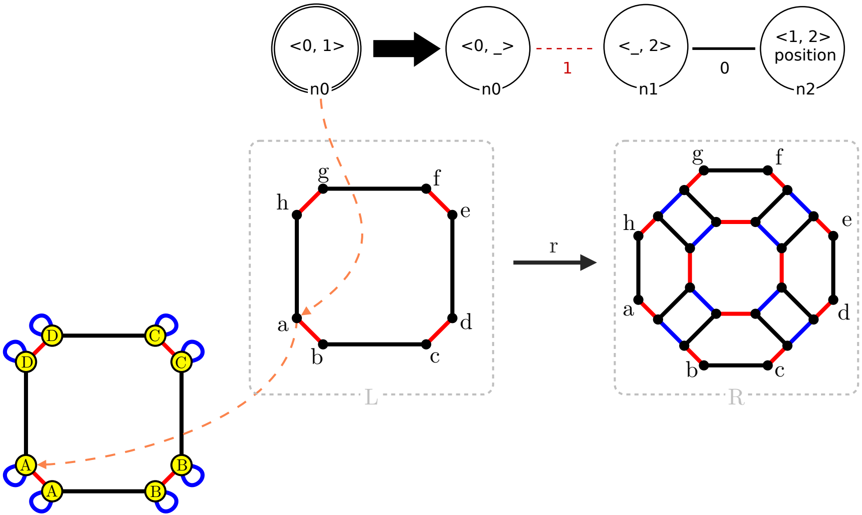
[1]Bellet et al. 2017.

Extension to schemes

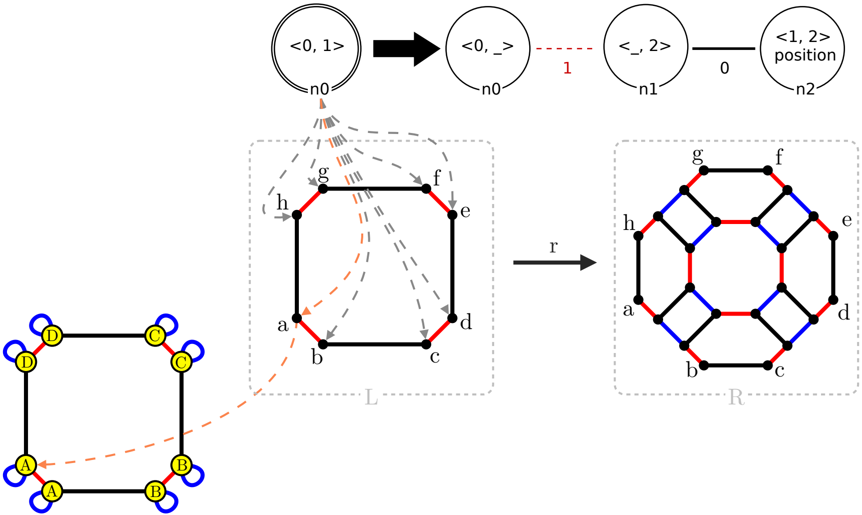




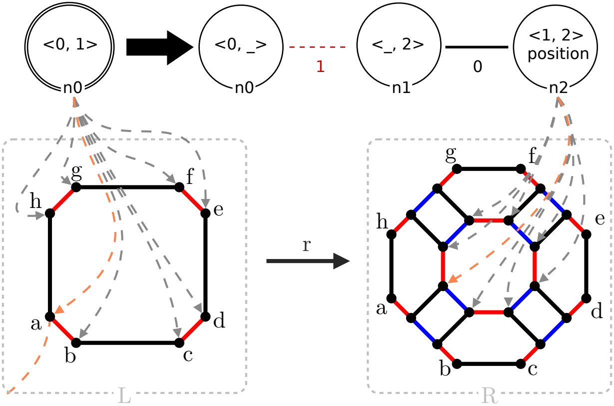
Extension to schemes

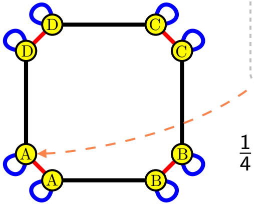


Extension to schemes



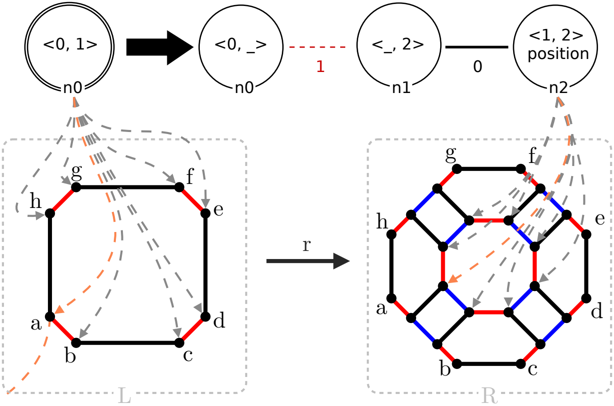
Extension to schemes

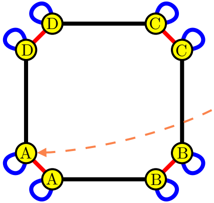




$\frac{1}{4}(A + B + C + D)$

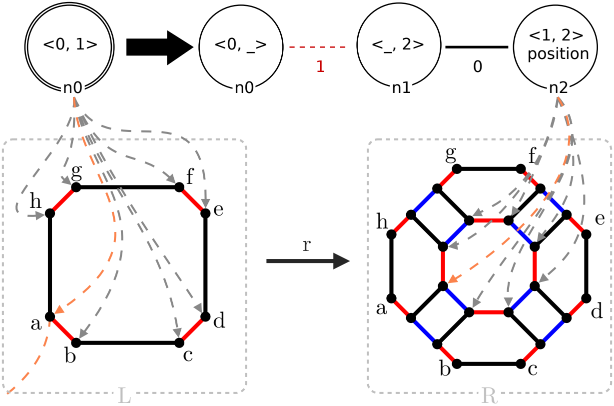
Extension to schemes

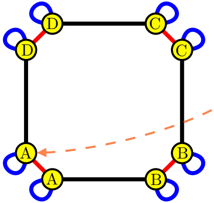




$$\frac{1}{4}(A + B + C + D) = \mathit{middle}(\{A, B, C, D\})$$

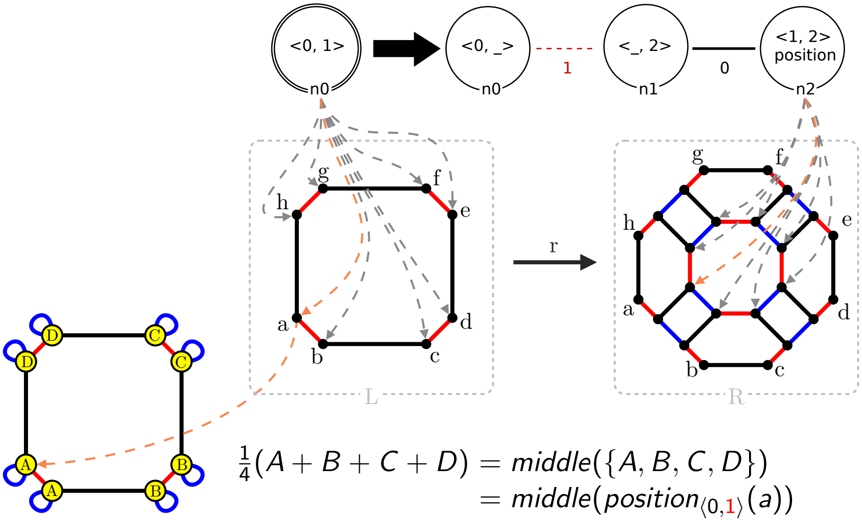
Extension to schemes





$$\begin{aligned}\tfrac{1}{4}(A + B + C + D) &= middle(\{A, B, C, D\}) \\ &= middle(position_{\langle 0,1\rangle}(a))\end{aligned}$$

Extension to schemes



$$\frac{1}{4}(A + B + C + D) = middle(\{A, B, C, D\})$$
$$= middle(position_{\langle 0, 1 \rangle}(a))$$
$$= middle(position_{\langle 0, 1 \rangle}(n0))$$

Consistency preservation

Modifications of a well-formed object should produce an equally well-formed object.

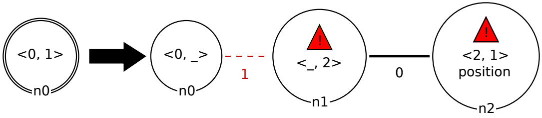
Requirement: Feedback to the rule designer.

Consistency preservation

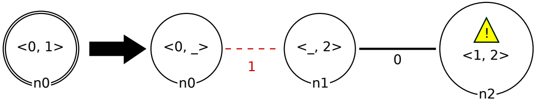
Modifications of a well-formed object should produce an equally well-formed object.

Requirement: Feedback to the rule designer.

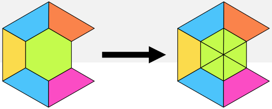
- Topological inconsistencies



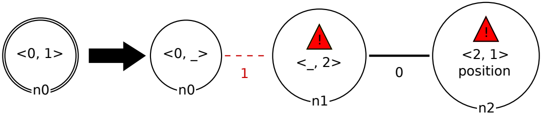
- Geometric inconsistencies

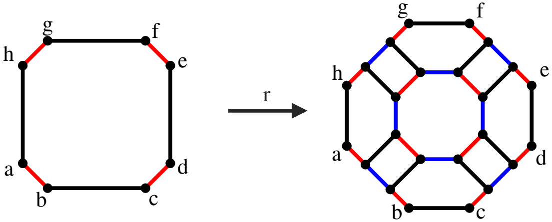


Breaking the topological consistency

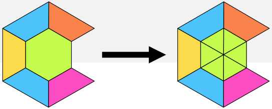


Constraint: 0202 paths should be cycles.

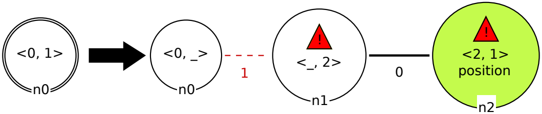


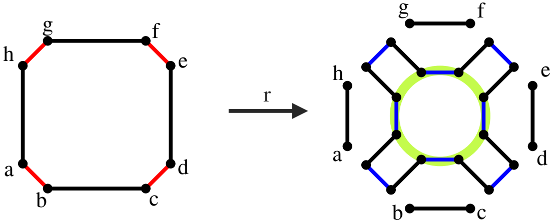


Breaking the topological consistency

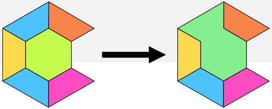


Constraint: 0202 paths should be cycles.

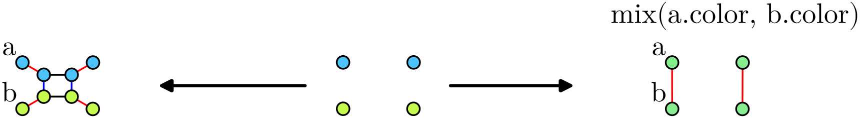




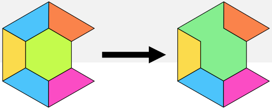
Breaking the geometric consistency



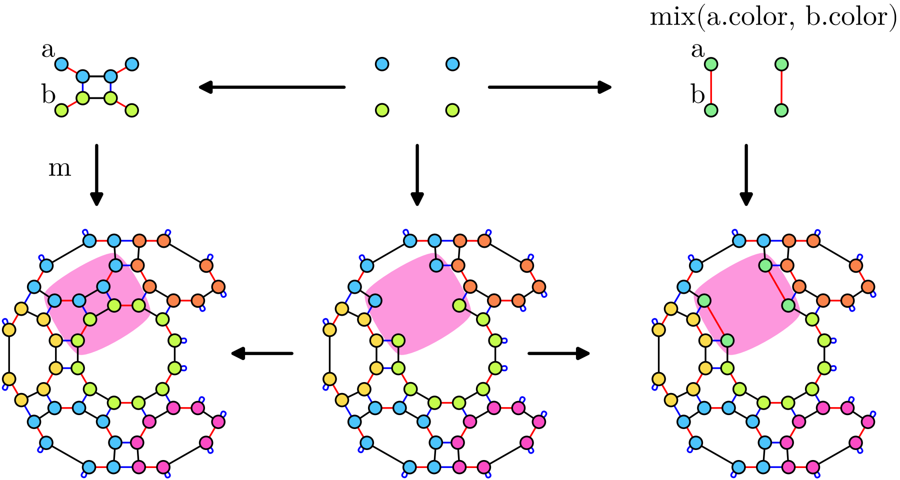
Constraint: nodes in a $\langle 0, 1 \rangle$-orbit should have the same color.



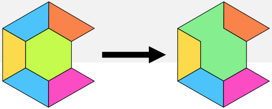
Breaking the geometric consistency



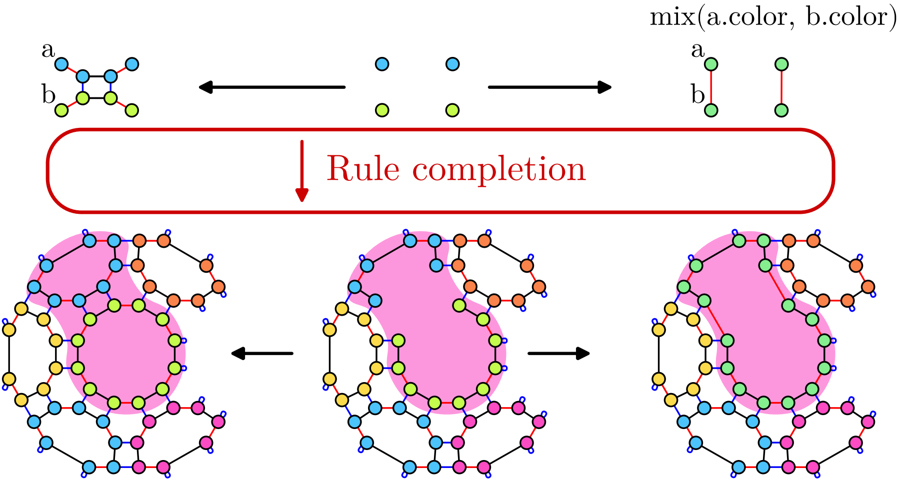
Constraint: nodes in a $\langle 0, 1 \rangle$-orbit should have the same color.



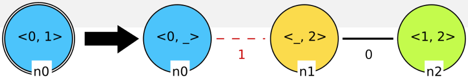
Breaking the geometric consistency



Constraint: nodes in a $\langle 0, 1 \rangle$-orbit should have the same color.



Formalizing Jerboa's DSL



- Jerboa's DSL with categorical constructions: products, attributes, completion.

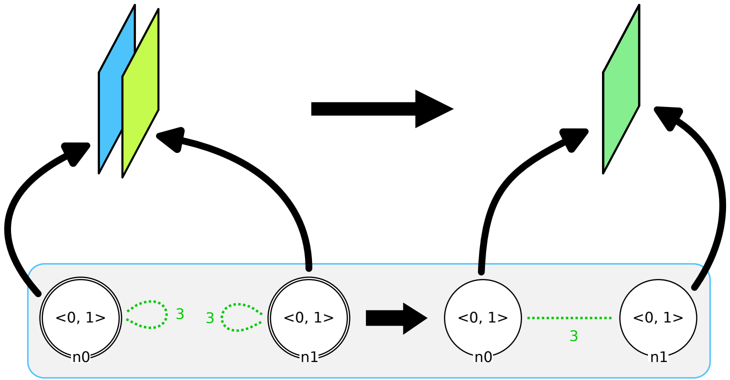
- Weaker consistency conditions.

Unified framework to study generalized and oriented maps.

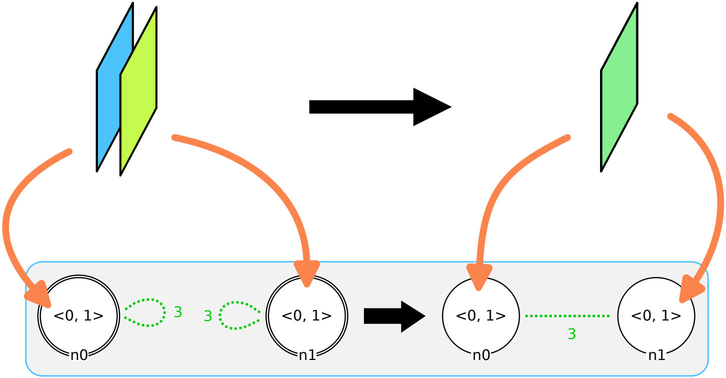
- Romain Pascual et al. (2022b). "Topological consistency preservation with graph transformation schemes". In: Science of Computer Programming. DOI: 10.1016/j.scico.2021.102728
- Agnès Arnould et al. (2022). "Preserving consistency in geometric modeling with graph transformations". In: Mathematical Structures in Computer Science. DOI: 10.1017/S0960129522000226

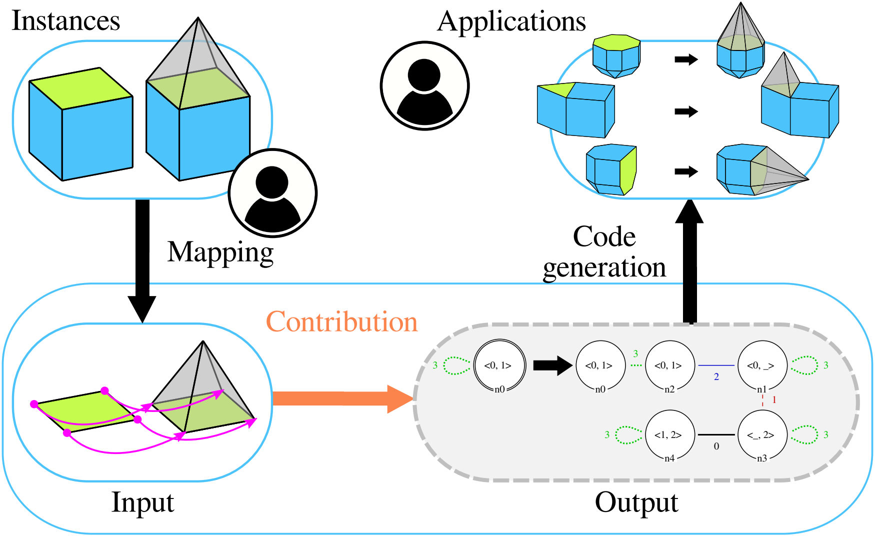
Inferring geometric modeling operations

- Retrieving the operation described by an example.

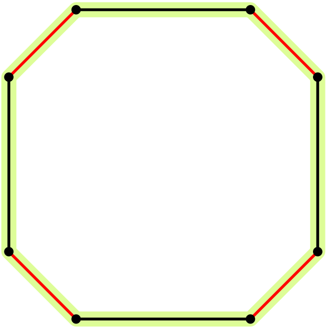


<0, 1>
n0
3
3
<0, 1>
n1
<0, 1>
n0
3
<0, 1>
n1

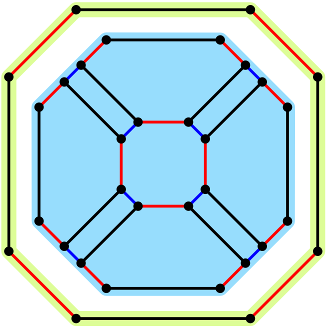

<0, 1>
n0
3
3
<0, 1>
n1
<0, 1>
n0
3
<0, 1>
n1


Instances
Applications
Mapping
Code generation
Contribution
<0, 1>
n0
3
<0, 1>
n0
3
<0, 1>
n2
2
<0, _>
n1
3
1
3
<1, 2>
n4
0
<_, 2>
n3
3
Input
Output

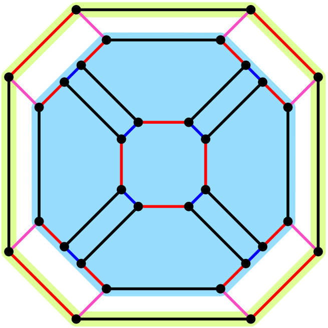
Folding a joint representation of the rule



Folding a joint representation of the rule

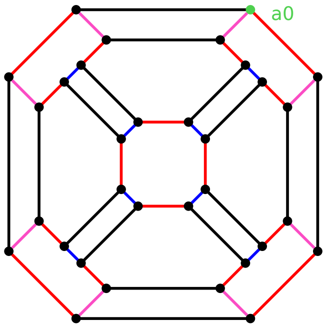


Folding a joint representation of the rule



Color legend: 0, 1, 2, κ.

Folding a joint representation of the rule



Color legend: 0, 1, 2, κ.

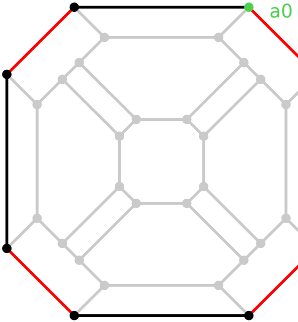
Topological folding algorithm: graph traversal folding nodes and arcs.

Input:

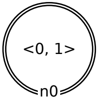
- two partial Gmaps
- preservation links
- a dart
- an orbit type.

- Orbit type $\langle 0, 1 \rangle$ and dart $a0$.

Execution

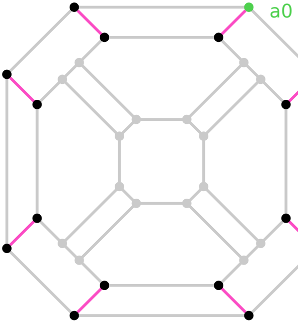


Hook (orbit $\langle 0, 1 \rangle$).

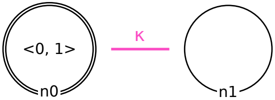


Color legend: 0, 1, 2, κ.

Execution

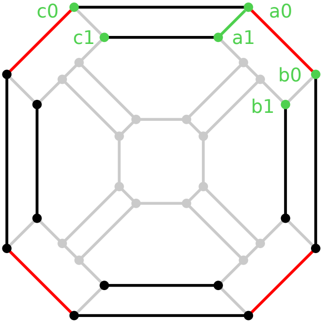


Folding the arcs.

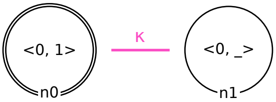


Color legend: 0, 1, 2, κ.

Execution

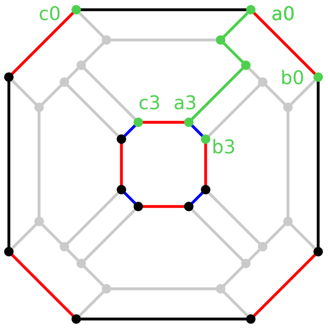


Folding a node.



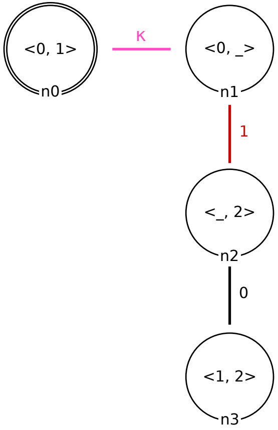
Color legend: 0, 1, 2, κ.

Execution

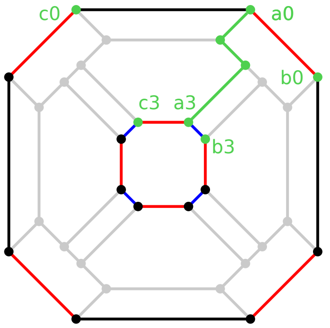


Color legend: 0, 1, 2, κ.

The algorithm terminates.

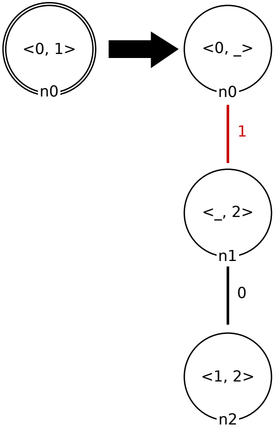


Execution



Color legend: 0, 1, 2, κ.

Splitting the joint representation.

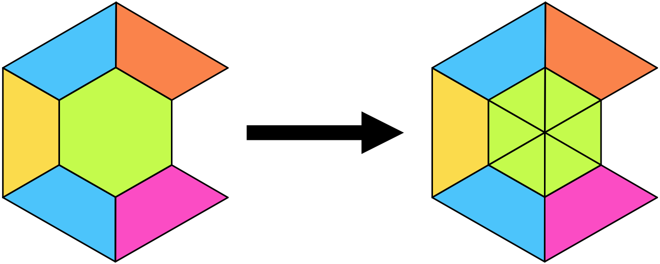


Results

Correctness: The algorithm returns a topological folding of the rule if it exists and halts otherwise.

- Romain Pascual et al. (2022a). "Inferring topological operations on generalized maps: Application to subdivision schemes". In: Graphics and Visual Computing. DOI: 10.1016/j.gvc.2022.200049

▶ What about cases where we cannot fold the rule? Orbit $\langle 0, 1, 2 \rangle$.

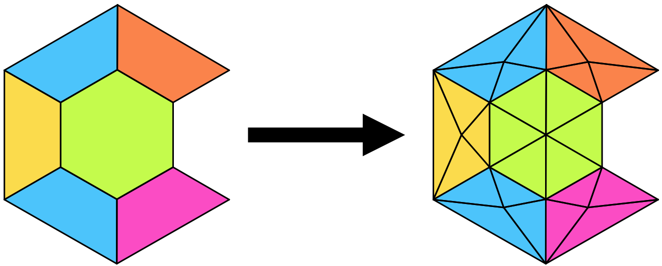


Results

Correctness: The algorithm returns a topological folding of the rule if it exists and halts otherwise.

- Romain Pascual et al. (2022a). "Inferring topological operations on generalized maps: Application to subdivision schemes". In: Graphics and Visual Computing. DOI: 10.1016/j.gvc.2022.200049

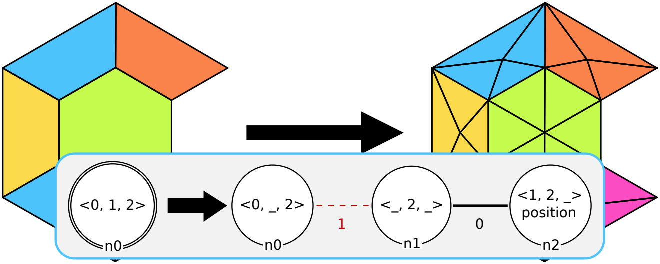
▶ What about cases where we cannot fold the rule? Orbit $\langle 0, 1, 2 \rangle$.



Correctness: The algorithm returns a topological folding of the rule if it exists and halts otherwise.

- Romain Pascual et al. (2022a). "Inferring topological operations on generalized maps: Application to subdivision schemes". In: Graphics and Visual Computing. DOI: 10.1016/j.gvc.2022.200049

▶ What about cases where we cannot fold the rule? Orbit $\langle 0, 1, 2 \rangle$.



Method (inference of positions)

Hypothesis: Affine combinations of positions.

Method (inference of positions)

Hypothesis: Affine combinations of positions.

For each vertex in C, we want a position p expressed as:

$$p = \sum_{i=0}^{k} w_i p_i + t$$

where:

- p : target position (known)

Method (inference of positions)

Hypothesis: Affine combinations of positions.

For each vertex in C, we want a position p expressed as:

$$p = \sum_{i=0}^{k} w_i p_i + t$$

where:

- p : target position (known)
- p_i: position of the initial vertex i (known)

Method (inference of positions)

Hypothesis: Affine combinations of positions.

For each vertex in C, we want a position p expressed as:

$$p = \sum_{i=0}^{k} w_i p_i + t$$

where:

- p : target position (known)
- p_i: position of the initial vertex i (known)
- w_i: weight (unknown)

Method (inference of positions)

Hypothesis: Affine combinations of positions.

For each vertex in C, we want a position p expressed as:

$$p = \sum_{i=0}^{k} w_i p_i + t$$

where:

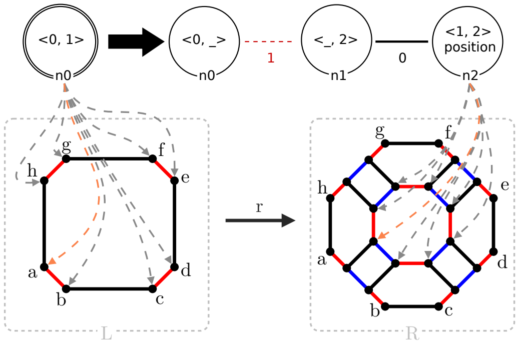
- p : target position (known)
- p_i: position of the initial vertex i (known)
- w_i: weight (unknown)
- t : translation (unknown)

Need for abstraction on schemes

$(w_i)_{0\leq i\leq k}$ such that: $p = \sum_{i=0}^{k} w_i p_i + t$

Need for abstraction on schemes

$(w_i)_{0 \leq i \leq k}$ such that: $p = \sum_{i=0}^{k} w_i p_i + t$

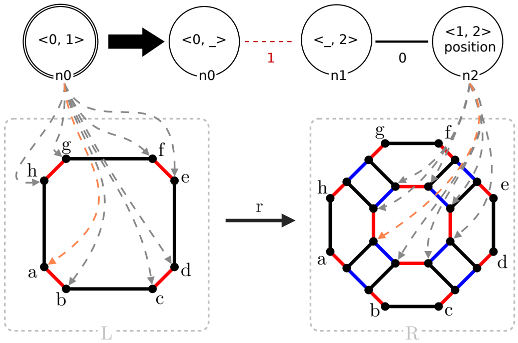


Issue: Darts share the same expression.

- Rule schemes abstract topological cells.

Need for abstraction on schemes

$(w_i)_{0 \leq i \leq k}$ such that: $p = \sum_{i=0}^{k} w_i p_i + t$



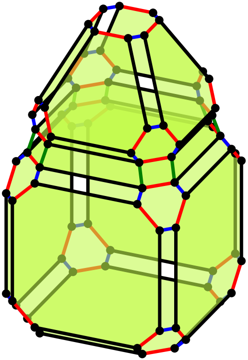
Issue: Darts share the same expression.

- Rule schemes abstract topological cells.

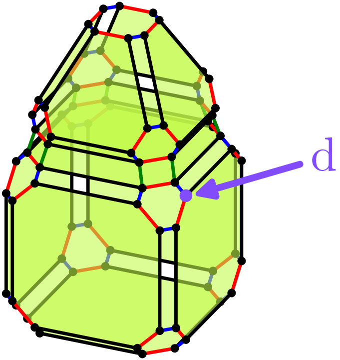
Solution: Exploit the topology.

- Use points of interest.

Points of interest



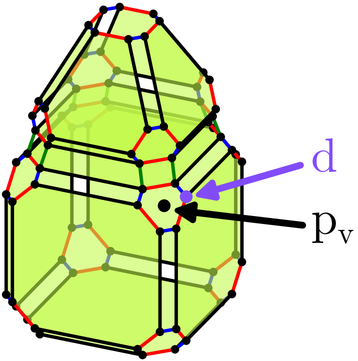
Points of interest



Points of interest

with

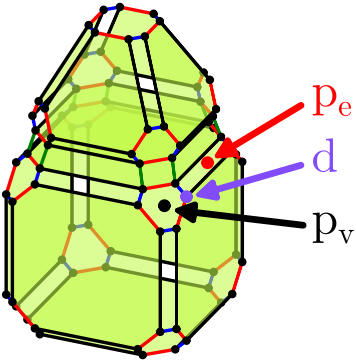
- p_v : vertex



$$p_v = middle(position_{\langle 1,2,3 \rangle}(d))$$

with

- p_v : vertex
- p_e : edge midpoint

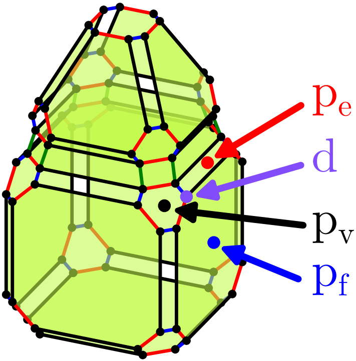


$$p_e = middle(position_{\langle 0,2,3\rangle}(d))$$

Points of interest

with

- p_v : vertex
- p_e : edge midpoint
- p_f : face barycenter

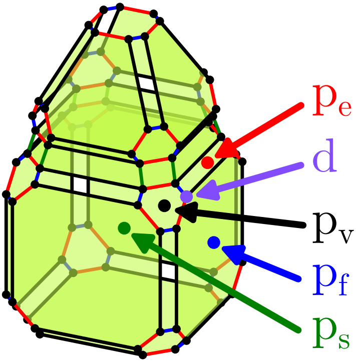


$$p_f = middle(position_{\langle 0,1,3 \rangle}(d))$$

Points of interest

with

- p_v : vertex
- p_e : edge midpoint
- p_f : face barycenter
- p_s : volume barycenter

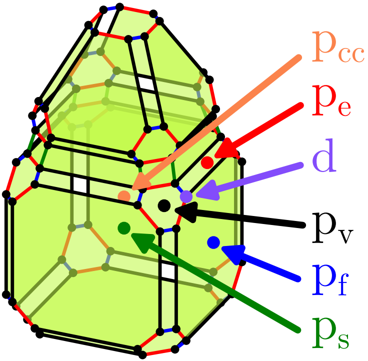


$$p_s = middle(position_{\langle 0,1,2\rangle}(d))$$

Points of interest

with

- p_v : vertex
- p_e : edge midpoint
- p_f : face barycenter
- p_s : volume barycenter
- p_{cc} : CC barycenter

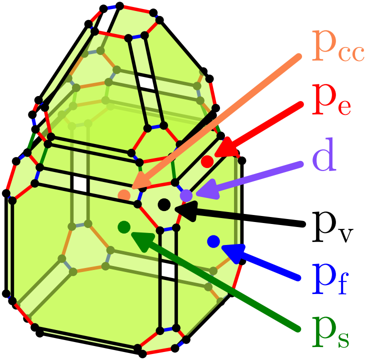


$$p_{cc} = middle(position_{\langle 0,1,2,3\rangle}(d))$$

Points of interest

with

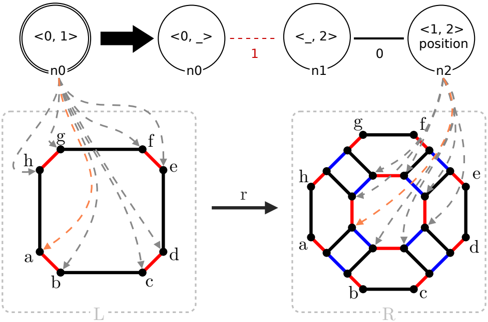
- p_v : vertex
- p_e : edge midpoint
- p_f : face barycenter
- p_s : volume barycenter
- p_{cc} : CC barycenter



The system is rewritten as:

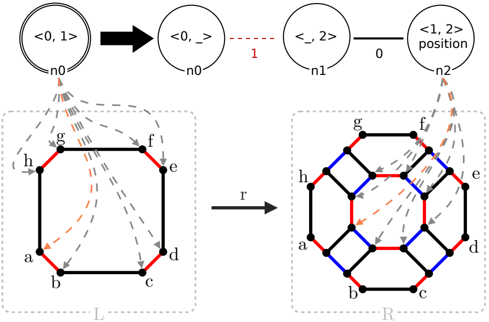
$$p = w_v p_v + w_e p_e + w_f p_f + w_s p_s + w_{cc} p_{cc} + t$$

Illustration



The position expression of $n2$ only depends on $n0$.

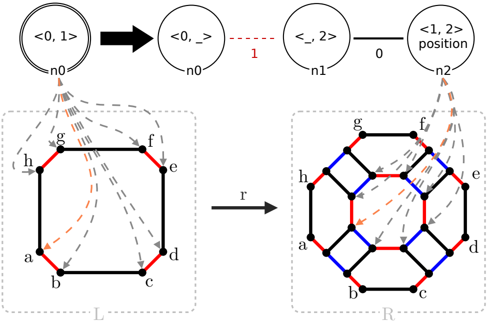
Illustration



The position expression of $n2$ only depends on $n0$.

$$n2.position = \underbrace{w_v\, n0.p_v}_{vertex} + \underbrace{w_e\, n0.p_e}_{edge} + \underbrace{w_f\, n0.p_f}_{face} + \underbrace{w_s\, n0.p_s}_{volume} + \underbrace{w_{cc}\, n0.p_{cc}}_{cc} + t$$

Illustration

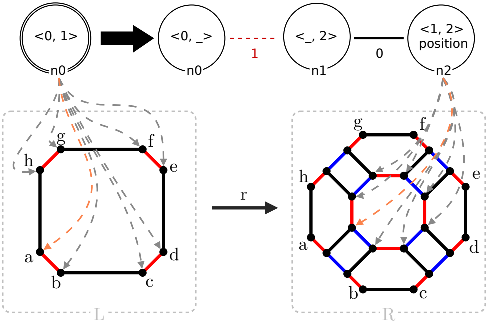


The position expression of $n2$ only depends on $n0$.

- One equation per dart (8 darts).

$$n2.position = \underbrace{w_v\, n0.p_v}_{vertex} + \underbrace{w_e\, n0.p_e}_{edge} + \underbrace{w_f\, n0.p_f}_{face} + \underbrace{w_s\, n0.p_s}_{volume} + \underbrace{w_{cc}\, n0.p_{cc}}_{cc} + t$$

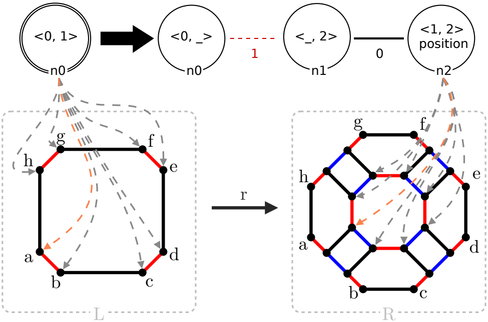
Illustration



The position expression of $n2$ only depends on $n0$.

- One equation per dart (8 darts).
- Split per coordinate (on x, y, z).

$$n2.position = \underbrace{w_v\, n0.p_v}_{vertex} + \underbrace{w_e\, n0.p_e}_{edge} + \underbrace{w_f\, n0.p_f}_{face} + \underbrace{w_s\, n0.p_s}_{volume} + \underbrace{w_{cc}\, n0.p_{cc}}_{cc} + t$$

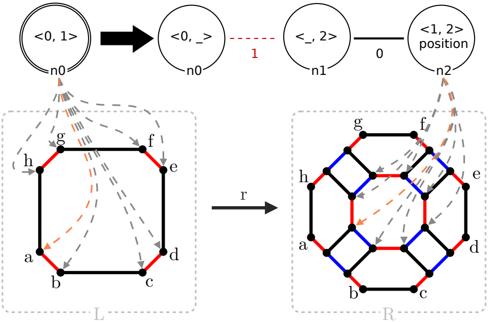


The position expression of $n2$ only depends on $n0$.

- One equation per dart (8 darts).
- Split per coordinate (on x, y, z).
- 24 equations and 8 variables.

$$n2.position = \underbrace{w_v\, n0.p_v}_{vertex} + \underbrace{w_e\, n0.p_e}_{edge} + \underbrace{w_f\, n0.p_f}_{face} + \underbrace{w_s\, n0.p_s}_{volume} + \underbrace{w_{cc}\, n0.p_{cc}}_{cc} + t$$

Illustration



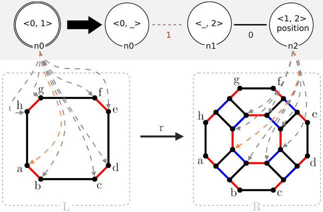
The position expression of $n2$ only depends on $n0$.

- One equation per dart (8 darts).
- Split per coordinate (on x, y, z).
- 24 equations and 8 variables.

$$n2.position = \underbrace{w_v\, n0.p_v}_{vertex} + \underbrace{w_e\, n0.p_e}_{edge} + \underbrace{w_f\, n0.p_f}_{face} + \underbrace{w_s\, n0.p_s}_{volume} + \underbrace{w_{cc}\, n0.p_{cc}}_{cc} + t$$

- Solved as a CSP. Solvers used: OR-Tools (Google), Z3 (Microsoft)

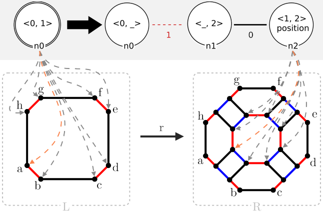
Solving the barycentric triangulation



- Global equation:

$$n2.position = w_v\, n0.p_v + w_e\, n0.p_e + w_f\, n0.p_f + w_s\, n0.p_s + w_{cc}\, n0.p_{cc} + t$$

Solving the barycentric triangulation



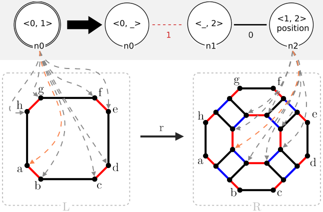
- Global equation:

$$n2.position = w_v\, n0.p_v + w_e\, {\color{red} n0.p_e} + w_f\, {\color{blue} n0.p_f} + w_s\, {\color{green} n0.p_s} + w_{cc}\, {\color{orange} n0.p_{cc}} + t$$

- Generated system

$$\begin{cases} (0.5;0.5) = w_v * (0;0) + w_e * {\color{red}(0.5;0)} + w_f * {\color{blue}(0.5;0.5)} + w_s * {\color{green}(0.5;0.5)} + w_{cc} * {\color{orange}(0.5;0.5)} + (tx;ty) \\ (0.5;0.5) = w_v * (1;0) + w_e * {\color{red}(0.5;0)} + w_f * {\color{blue}(0.5;0.5)} + w_s * {\color{green}(0.5;0.5)} + w_{cc} * {\color{orange}(0.5;0.5)} + (tx;ty) \\ (0.5;0.5) = w_v * (1;0) + w_e * {\color{red}(1;0.5)} + w_f * {\color{blue}(0.5;0.5)} + w_s * {\color{green}(0.5;0.5)} + w_{cc} * {\color{orange}(0.5;0.5)} + (tx;ty) \\ (0.5;0.5) = w_v * (1;1) + w_e * {\color{red}(1;0.5)} + w_f * {\color{blue}(0.5;0.5)} + w_s * {\color{green}(0.5;0.5)} + w_{cc} * {\color{orange}(0.5;0.5)} + (tx;ty) \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \end{cases}$$

Solving the barycentric triangulation



- Global equation:

$$n2.position = w_v\, n0.p_v + w_e\, n0.p_e + w_f\, n0.p_f + w_s\, n0.p_s + w_{cc}\, n0.p_{cc} + t$$

- Generated system

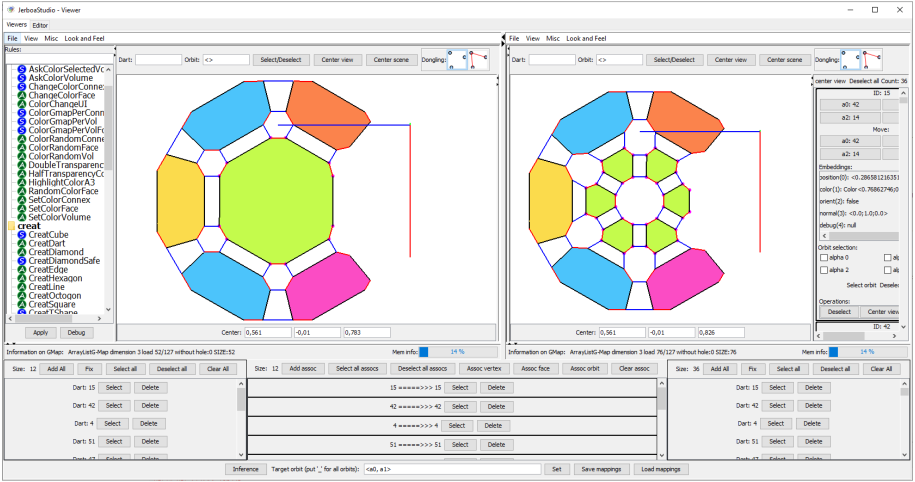
$$\begin{cases} (0.5;0.5) = w_v * (0;0) + w_e * (0.5;0) + w_f * (0.5;0.5) + w_s * (0.5;0.5) + w_{cc} * (0.5;0.5) + (tx;ty) \\ (0.5;0.5) = w_v * (1;0) + w_e * (0.5;0) + w_f * (0.5;0.5) + w_s * (0.5;0.5) + w_{cc} * (0.5;0.5) + (tx;ty) \\ (0.5;0.5) = w_v * (1;0) + w_e * (1;0.5) + w_f * (0.5;0.5) + w_s * (0.5;0.5) + w_{cc} * (0.5;0.5) + (tx;ty) \\ (0.5;0.5) = w_v * (1;1) + w_e * (1;0.5) + w_f * (0.5;0.5) + w_s * (0.5;0.5) + w_{cc} * (0.5;0.5) + (tx;ty) \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \end{cases}$$

- Solution found:
  - $w_v = 0.0$
  - $w_e = 0.0$
  - $w_f = 1.0$
  - $w_s = 0.0$
  - $w_{cc} = 0.0$
  - $t = (0.0, 0.0)$

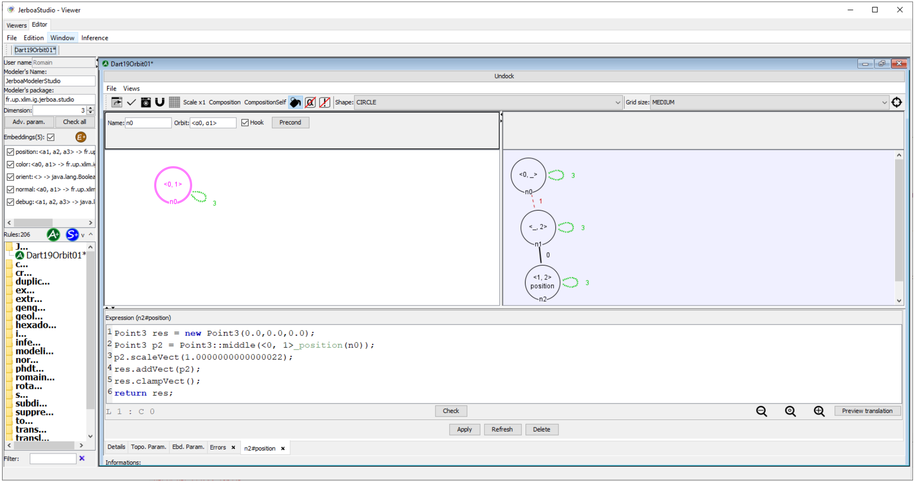
JerboaStudio and applications

- Implementation of the inference mechanism in Jerboa.

JerboaStudio

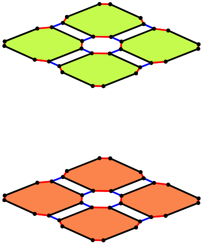


JerboaStudio

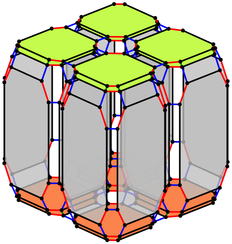


Example inspired from geology

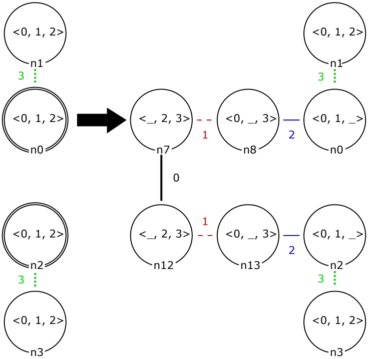
Before



After

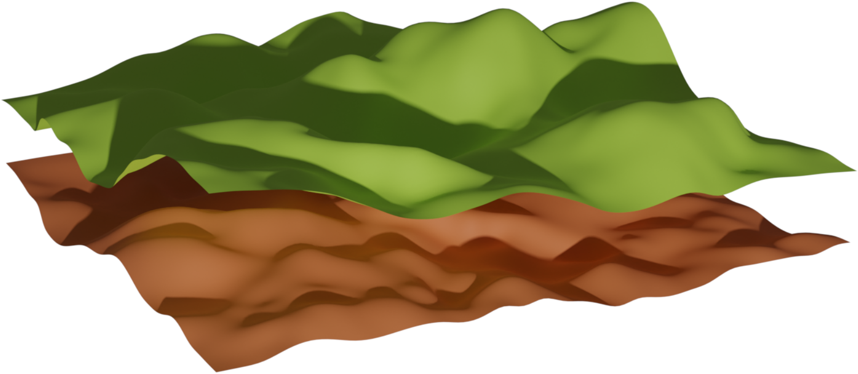


Operation

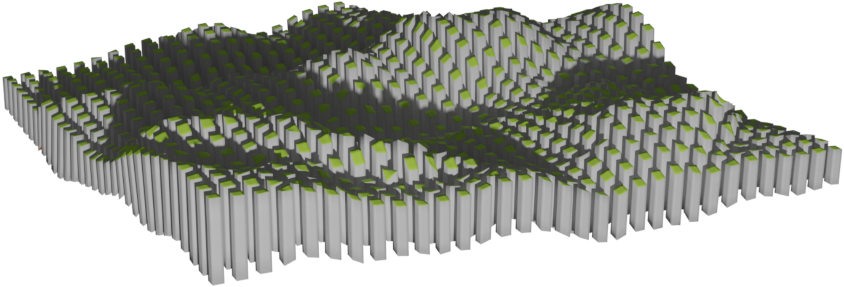


Inference time: ~ 3 ms

Example inspired from geology

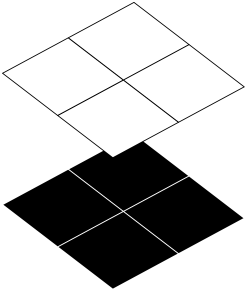


Example inspired from geology

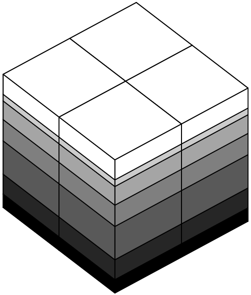


Example inspired from geology (part 2)

Before

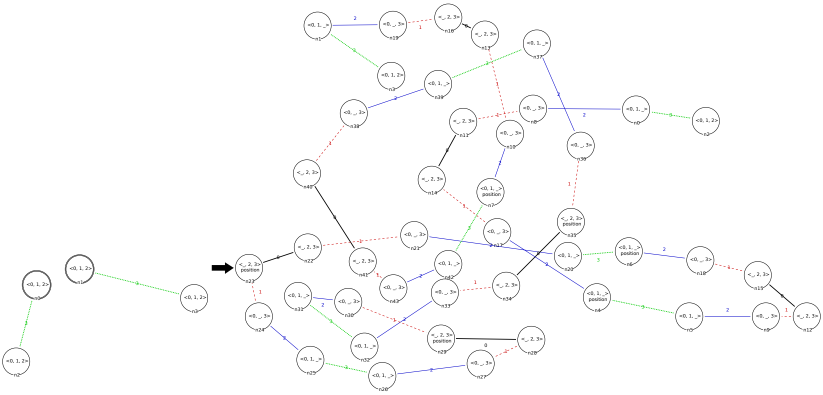


After



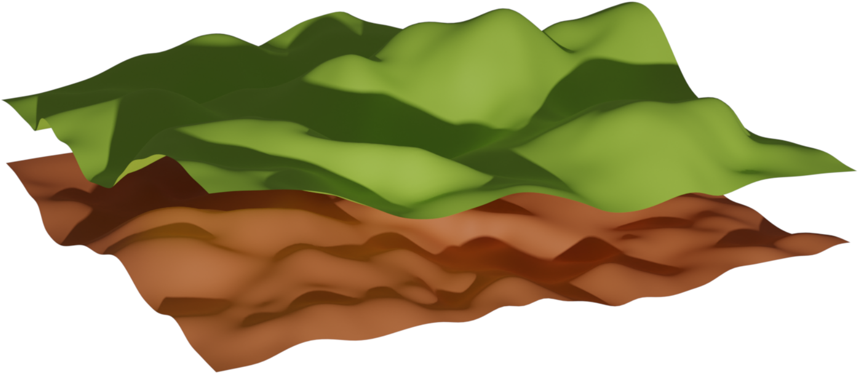
Both positions and colors.

Example inspired from geology (part 2)

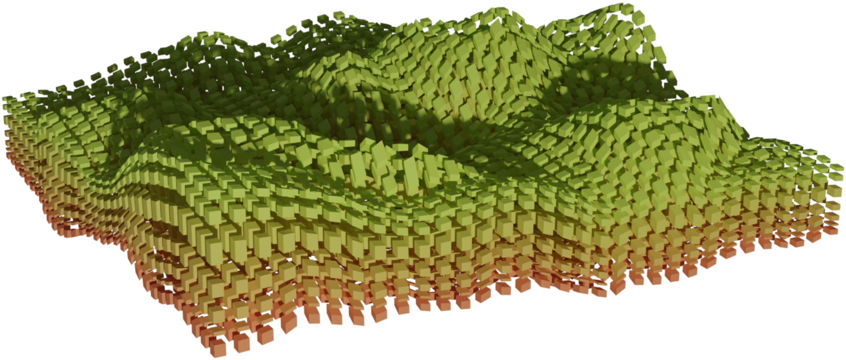


Inference time: $\sim$ 26 ms for the topology,
$\sim$ 549 ms for the embedding expressions

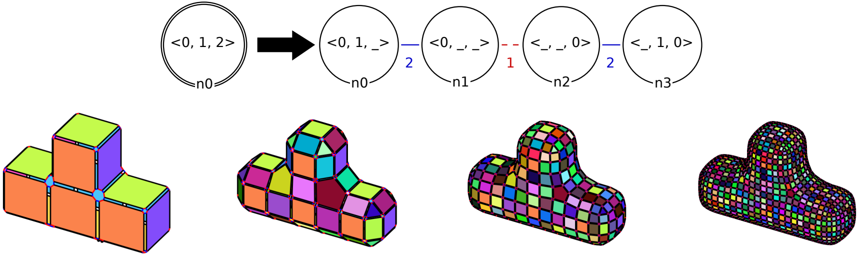
Example inspired from geology



Example inspired from geology

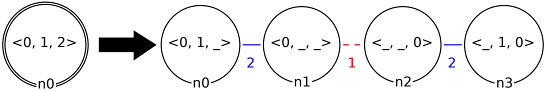


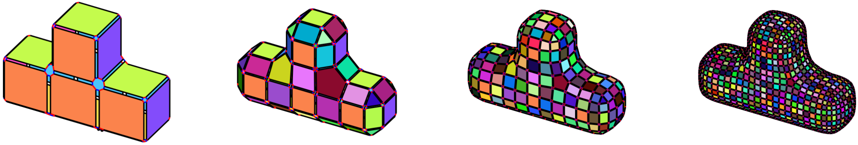
Doo-Sabin subdivision[1]



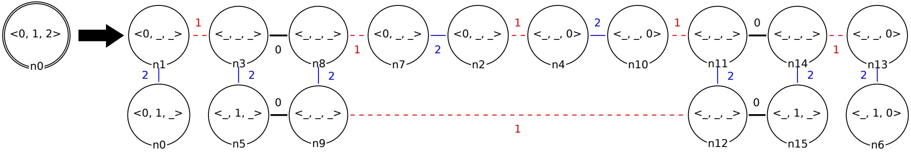
[1] Doo et al. 1978.

Doo-Sabin subdivision[1]



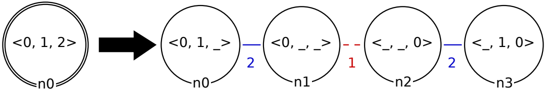


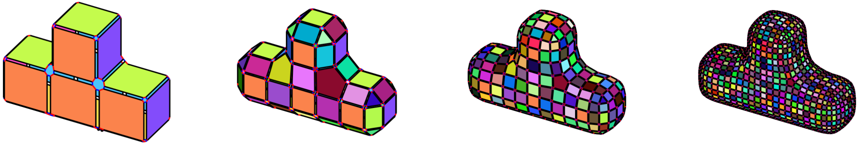
- 2^{nd} iteration:



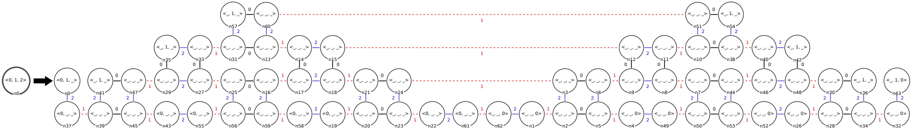
[1]Doo et al. 1978.

Doo-Sabin subdivision[1]



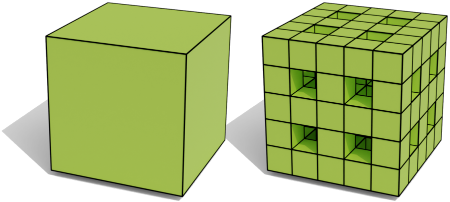


- 3^{rd} iteration:



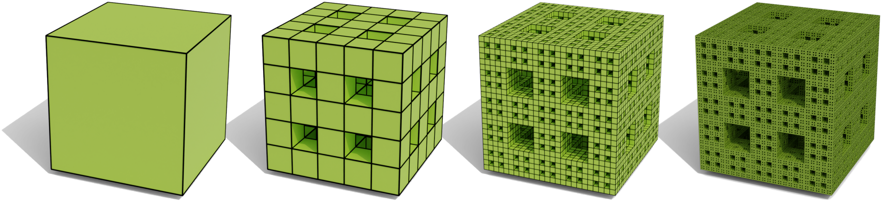
[1] Doo et al. 1978.

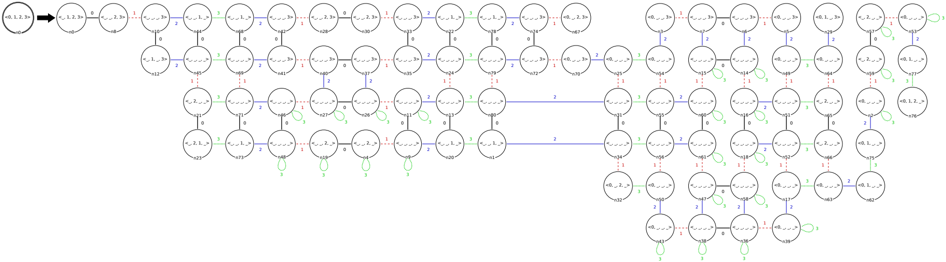
Menger $(2, 2, 2)$[1]



[1] Richaume et al. 2019.

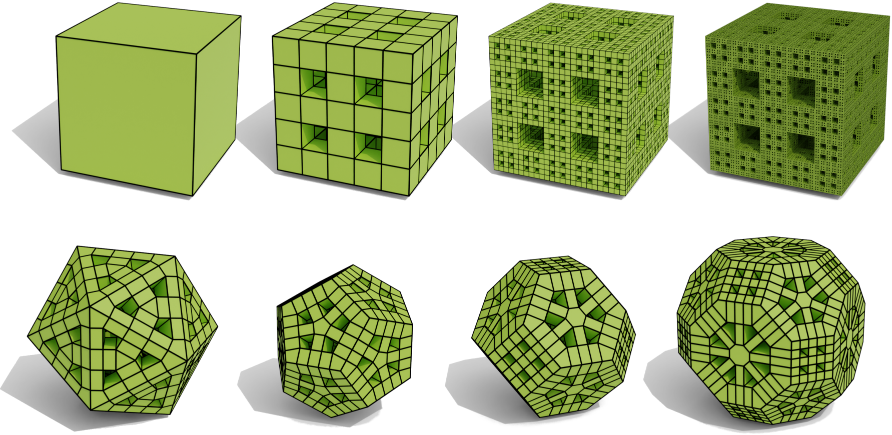
Menger $(2, 2, 2)$[1]





[1] Richaume et al. 2019.

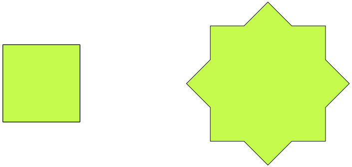
Menger $(2, 2, 2)$[1]



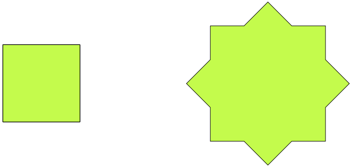
[1]Richaume et al. 2019.

Edge cases

- Von Koch's snowflake generated with L-systems

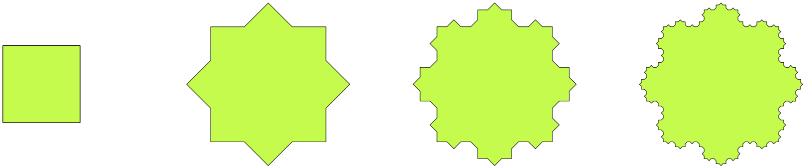


- Inferred:

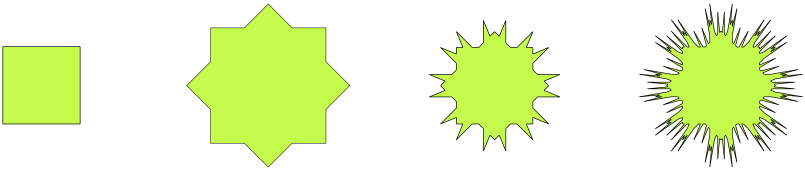


Edge cases

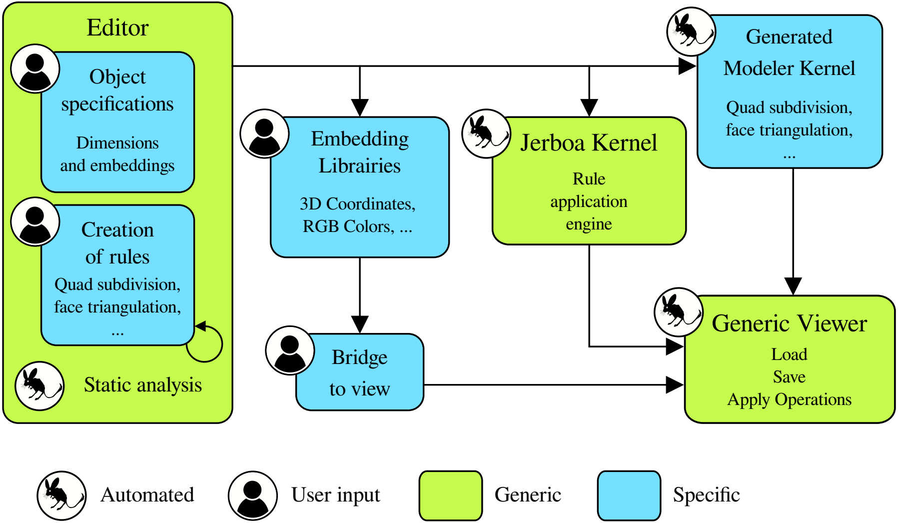
- Von Koch's snowflake generated with L-systems

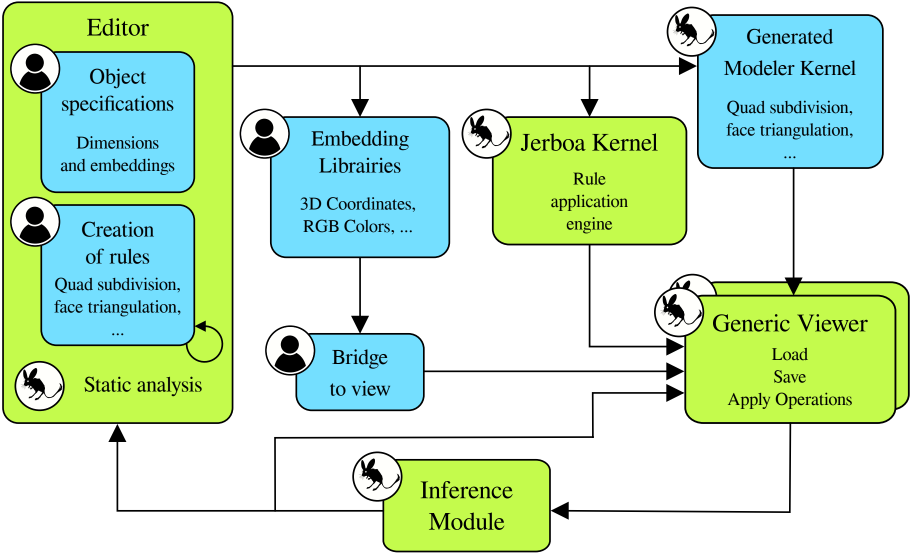


- Inferred:



JerboaStudio's architecture




Editor
Object specifications
Dimensions and embeddings
Creation of rules
Quad subdivision, face triangulation, ...
Static analysis
Embedding Librairies
3D Coordinates, RGB Colors, ...
Bridge to view
Jerboa Kernel
Rule application engine
Generated Modeler Kernel
Quad subdivision, face triangulation, ...
Generic Viewer
Load
Save
Apply Operations
Inference Module

Conclusion

- Ongoing works and main contributions.

Towards local nested conditions?

Rules should be checked statically.

Towards local nested conditions?

Rules should be checked statically.

Calculus for constraint preserving and constraint guaranteeing rules.[1]

[1] Pennemann 2009.

Towards local nested conditions?

Rules should be checked statically.

Calculus for constraint preserving and constraint guaranteeing rules.[1]

- Rely on global computations.
- Does not scale very well (with the size of the graphs).

[1] Pennemann 2009.

Towards local nested conditions?

Rules should be checked statically.

Calculus for constraint preserving and constraint guaranteeing rules.[1]

- Rely on global computations.
- Does not scale very well (with the size of the graphs).

▶ Collaboration with Nicolas Behr and Pascale Le Gall.

[1] Pennemann 2009.

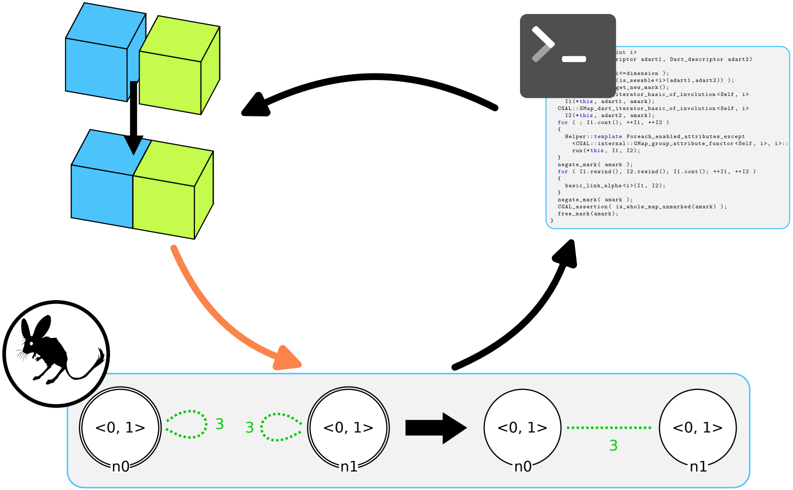
Towards a multi-cell query-replace approach?

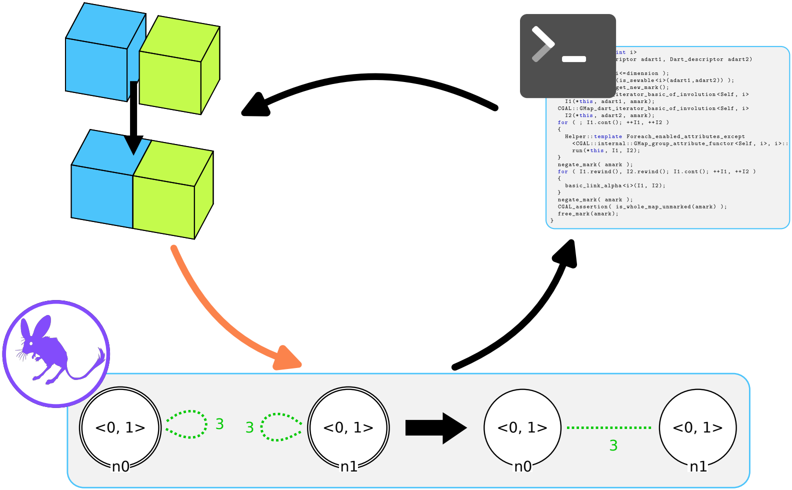
Query-replace modifications with a text editor but for combinatorial structures.[1]

How to extend the approach to multicell patterns?

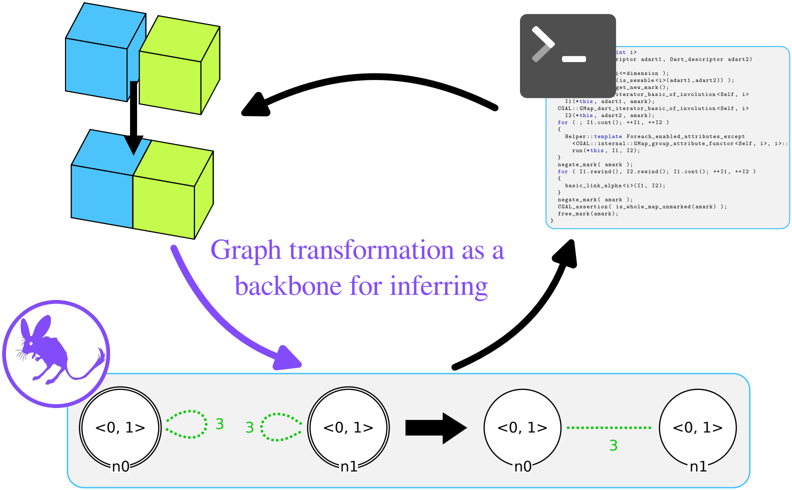
▶ Collaboration with Guillaume Damiand and Vincent Nivoliers supported by the GDR IGRV.

[1] Damiand et al. 2022


int i>
riptor adart1, Dart_descriptor adart2)
i<=dimension);
(is_sewable<i>(adart1,adart2)));
get_new_mark();
iterator_basic_of_involution<Self, i>
I1(*this, adart1, amark);
CGAL::GMap_dart_iterator_basic_of_involution<Self, i>
I2(*this, adart2, amark);
for (; I1.cont(); ++I1, ++I2)
{
Helper::template Foreach_enabled_attributes_except
<CGAL::internal::GMap_group_attribute_functor<Self, i>, i>::
run(*this, I1, I2);
}
negate_mark(amark);
for (I1.rewind(), I2.rewind(); I1.cont(); ++I1, ++I2)
{
basic_link_alpha<i>(I1, I2);
}
negate_mark(amark);
CGAL_assertion(is_whole_map_unmarked(amark));
free_mark(amark);
}
<0, 1>
n0
3
3
<0, 1>
n1
<0, 1>
n0
3
<0, 1>
n1

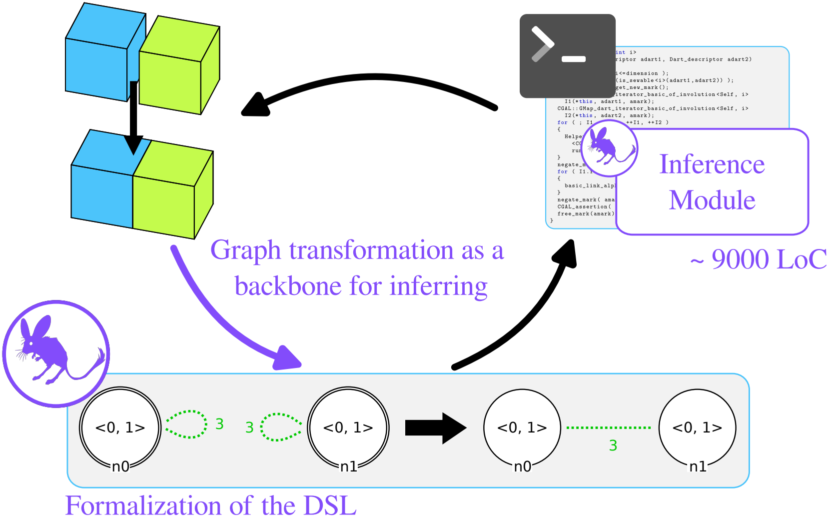


Formalization of the DSL



Graph transformation as a backbone for inferring

Formalization of the DSL


Graph transformation as a
backbone for inferring
Inference
Module
~ 9000 LoC
<0, 1>
n0
3
3
<0, 1>
n1
<0, 1>
n0
3
<0, 1>
n1
Formalization of the DSL

- Arnould, Agnès et al. (2022). "Preserving consistency in geometric modeling with graph transformations". In: Mathematical Structures in Computer Science. DOI: 10.1017/S0960129522000226.
- Bauderon, Michel (Jan. 1, 1995). "Parallel Rewriting of Graphs through the Pullback Approach". In: Electronic Notes in Theoretical Computer Science. SEGRAGRA 1995 2, pp. 19–26. ISSN: 1571-0661. DOI: 10.1016/S1571-0661(05)80176-8.
- Belhaouari, Hakim et al. (2014). "Jerboa: A Graph Transformation Library for Topology-Based Geometric Modeling". In: Graph Transformation. ICGT 2014. Ed. by Holger Giese et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 269–284. ISBN: 978-3-319-09108-2. DOI: 10.1007/978-3-319-09108-2_18.

References II

Bellet, Thomas et al. (2017). "Geometric Modeling: Consistency Preservation Using Two-Layered Variable Substitutions". In: Graph Transformation (ICGT 2017). Ed. by Juan de Lara et al. Vol. 10373. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 36–53. ISBN: 978-3-319-61470-0. DOI: 10.1007/978-3-319-61470-0_3.

Damiand, Guillaume et al. (Sept. 19, 2014). Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing. CRC Press. 407 pp. ISBN: 978-1-4822-0652-4.

Damiand, Guillaume et al. (June 18, 2022). "Query-replace operations for topologically controlled 3D mesh editing". In: Computers & Graphics. ISSN: 0097-8493. DOI: 10.1016/j.cag.2022.06.008.

References III

- Doo, Daniel et al. (Nov. 1, 1978). "Behaviour of recursive division surfaces near extraordinary points". In: Computer-Aided Design 10.6, pp. 356–360. ISSN: 0010-4485. DOI: 10.1016/0010-4485(78)90111-2.
- Ehrig, Hartmut et al. (2006). Fundamentals of Algebraic Graph Transformation. Monographs in Theoretical Computer Science. An EATCS Series. Berlin Heidelberg: Springer-Verlag. ISBN: 978-3-540-31187-4. DOI: 10.1007/3-540-31188-2.
- Heckel, Reiko et al. (2020). Graph Transformation for Software Engineers: With Applications to Model-Based Development and Domain-Specific Language Engineering. Cham: Springer International Publishing. ISBN: 978-3-030-43915-6. DOI: 10.1007/978-3-030-43916-3.

References IV

- Pascual, Romain et al. (2022a). "Inferring topological operations on generalized maps: Application to subdivision schemes". In: Graphics and Visual Computing. DOI: 10.1016/j.gvc.2022.200049.
- Pascual, Romain et al. (2022b). "Topological consistency preservation with graph transformation schemes". In: Science of Computer Programming. DOI: 10.1016/j.scico.2021.102728.
- Pennemann, Karl-Heinz (2009). "Development of correct graph transformation systems". PhD thesis. Department für Informatik, Universität Oldenburg, URL: http://oops.uni-oldenburg.de/884.
- Poudret, Mathieu et al. (2008). "Graph Transformation for Topology Modelling". In: Graph Transformations. ICGT 2008. Ed. by Hartmut Ehrig et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 147–161. DOI: 10.1007/978-3-540-87405-8_11.

References V

Richaume, Lydie et al. (2019). "Unfolding Level 1 Menger Polycubes of Arbitrary Size With Help of Outer Faces". In: Discrete Geometry for Computer Imagery. Ed. by Michel Couprie et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 457–468. ISBN: 978-3-030-14085-4. DOI: 10.1007/978-3-030-14085-4_36.

Rozenberg, Grzegorz, ed. (Feb. 1, 1997). Handbook of Graph Grammars and Computing by Graph Transformation: Volume I. Foundations. Vol. Foundations. 1 vols. USA: World Scientific Publishing Co., Inc. 545 pp. ISBN: 978-981-02-2884-2.