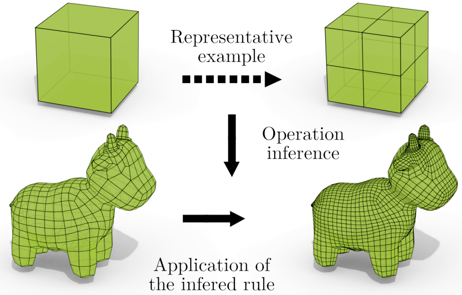
Inference of geometric modeling operations



Romain Pascual

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

March 3, 2023

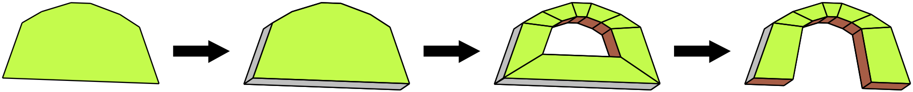
Geometric modeling

- How to realize such a scene?

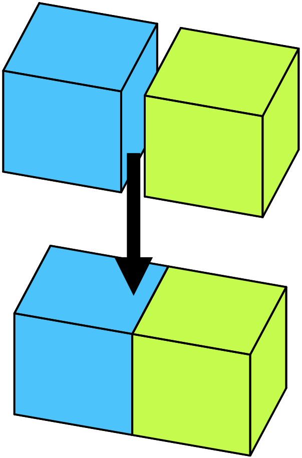
Geometric modeling

- How to realize such a scene?

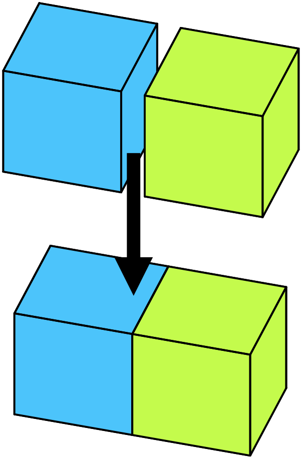
- Creating objects



Designing modeling operations



Designing modeling operations

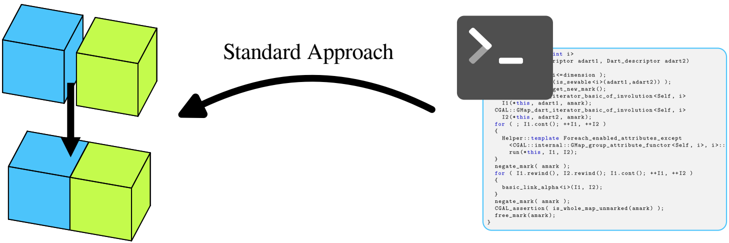


CGAL's sew operation

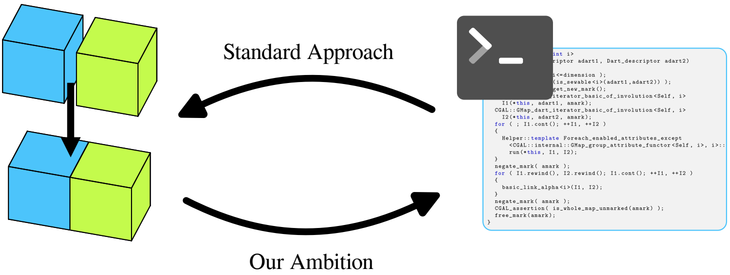
```
template<unsigned int i>
void sew(Dart_descriptor adart1, Dart_descriptor adart2)

  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);
}
```

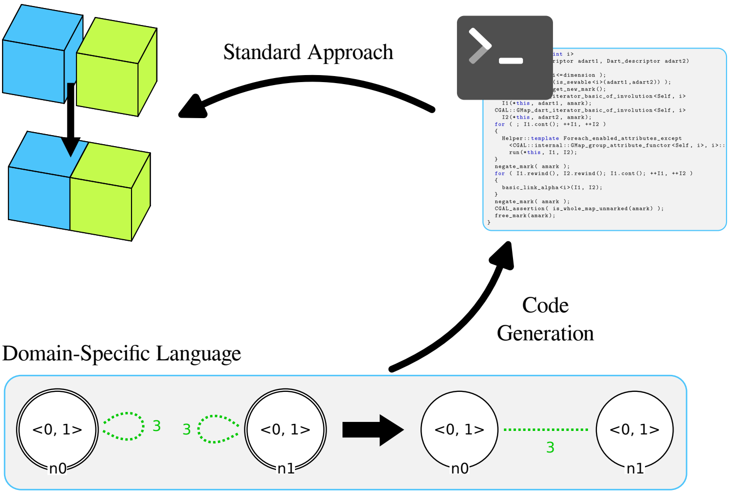
Inferring modeling operations



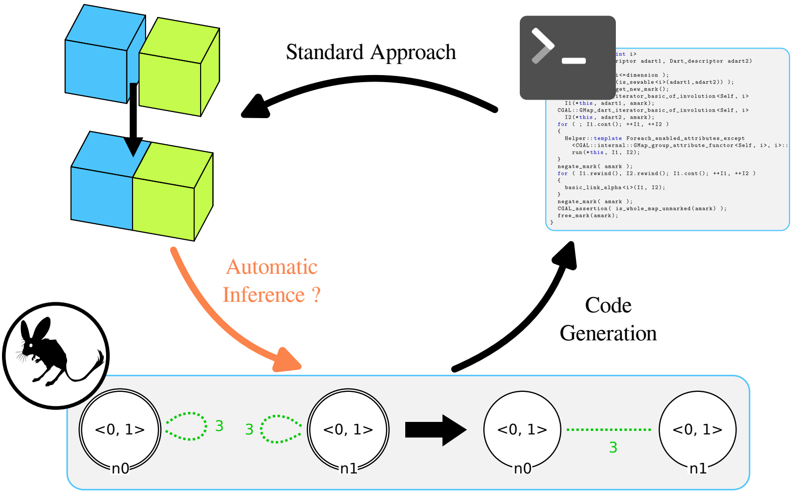
Inferring modeling operations



Inferring modeling operations

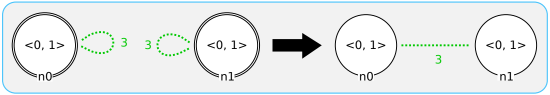


Inferring modeling operations

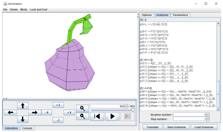
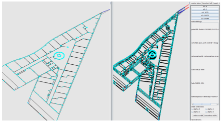
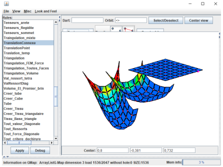
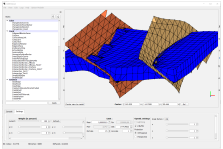


Strengths and weaknesses of Jerboa's DSL

- Main characteristics:
  - Dedicated to Gmaps[1]
  - Based on graph rewriting[2]

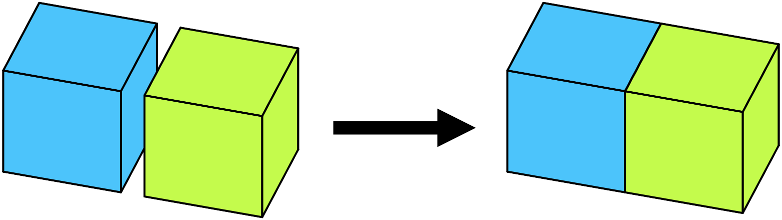


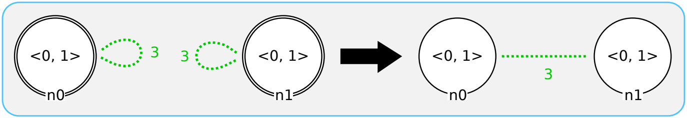
- Successful applications:
  - Plant growth
  - Architecture
  - Spring-mass
  - Geology

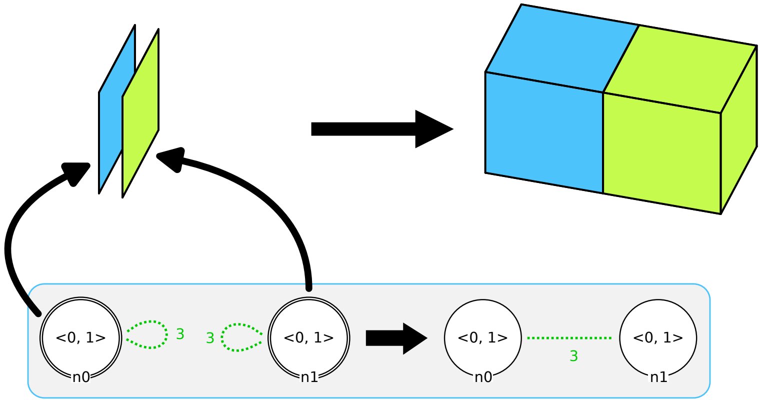
[1]Poudret et al. 2008.
[2]Belhaouari et al. 2014.

A sneak peek at Jerboa's language

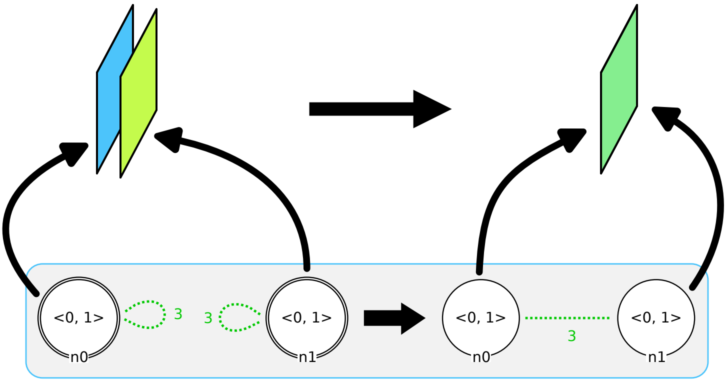




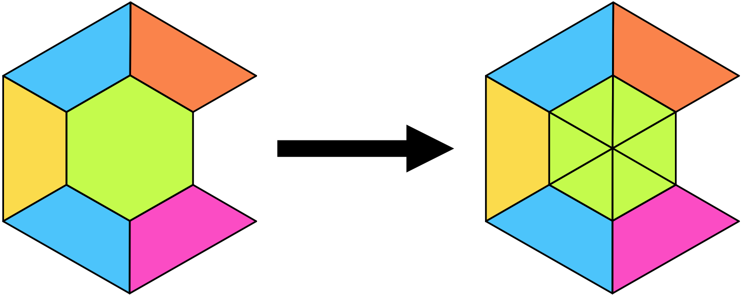
A sneak peek at Jerboa's language



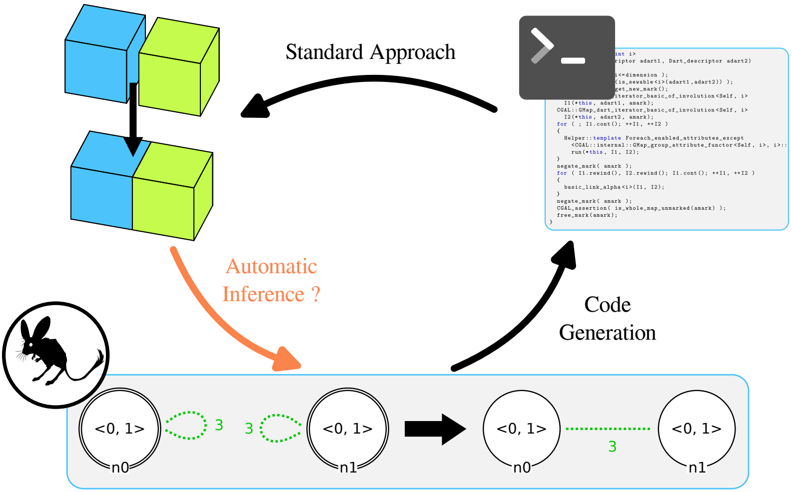
A sneak peek at Jerboa's language



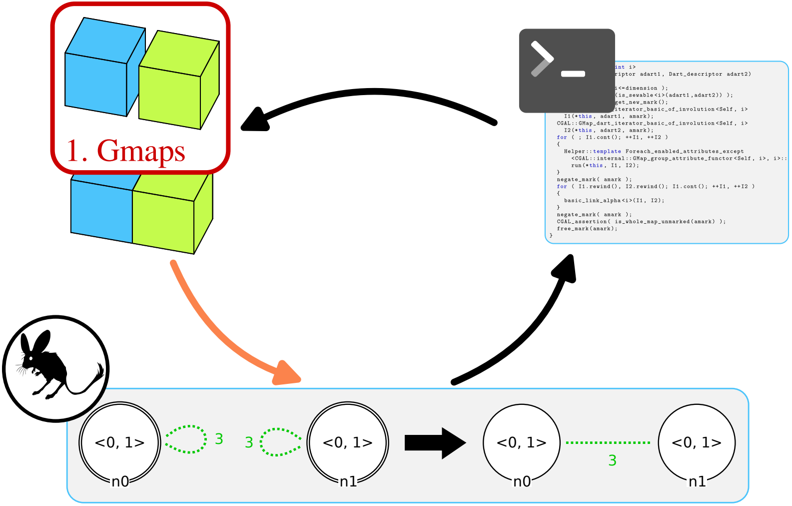
Running example: face triangulation



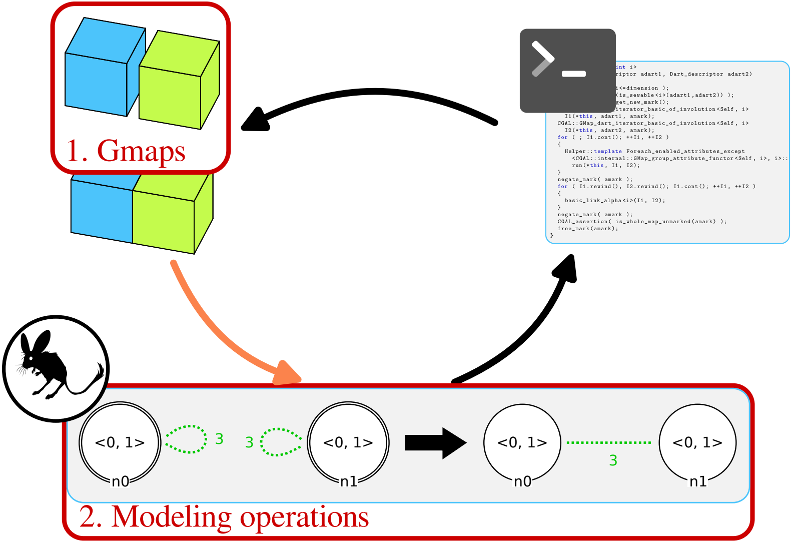
Plan


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);
}
Automatic Inference ?
Code Generation
<0, 1>
n0
3
3
<0, 1>
n1
<0, 1>
n0
3
<0, 1>
n1

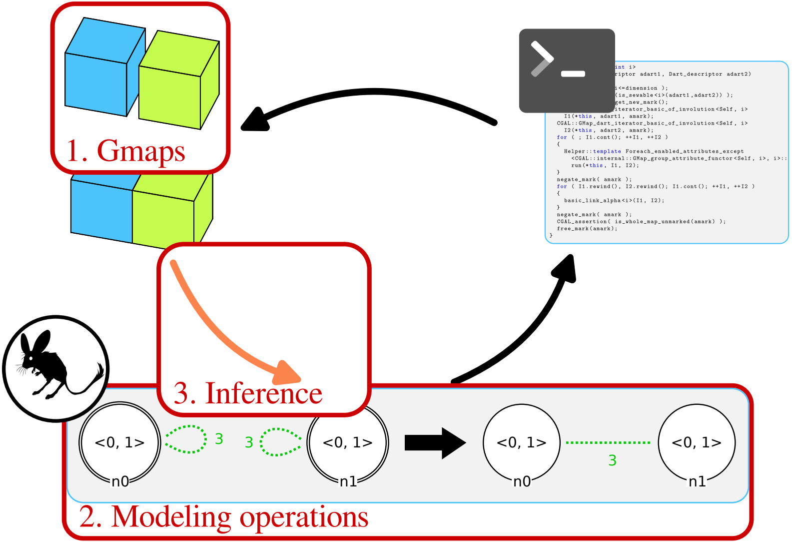
Plan



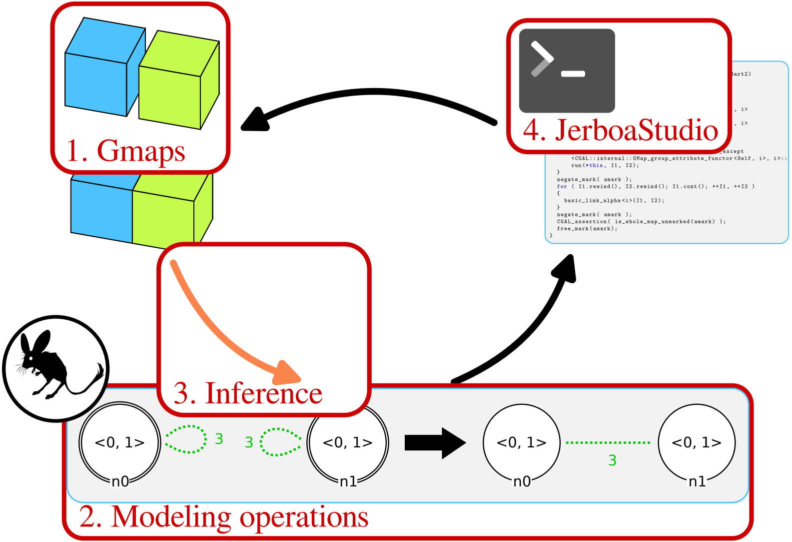
Plan



Plan


1. Gmaps
3. Inference
2. Modeling operations
<0, 1>
n0
3
3
<0, 1>
n1
<0, 1>
n0
3
<0, 1>
n1

Plan

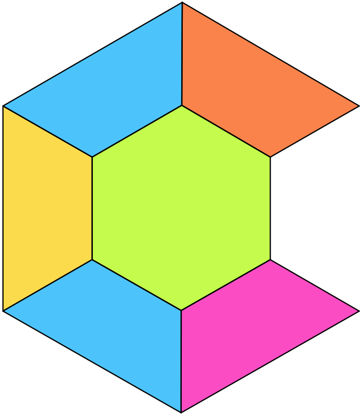


1. Gmaps
4. JerboaStudio
3. Inference
<0, 1>
n0
3
3
<0, 1>
n1
<0, 1>
n0
3
<0, 1>
n1
2. Modeling operations

Generalized maps

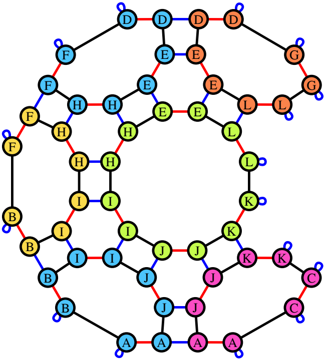
- Geometric objects are represented with embedded generalized maps.

Generalized maps[1]



[1]Damiand et al. 2014.

Generalized maps[1]

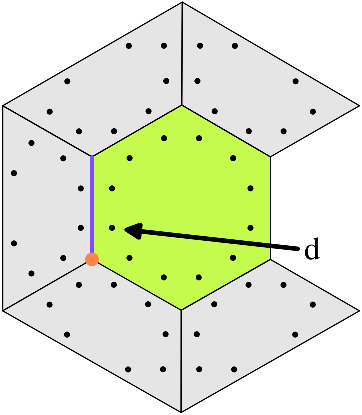


Color legend: 0, 1, 2.

Gmaps built as graphs.

[1] Damiand et al. 2014.

Generalized maps[1]



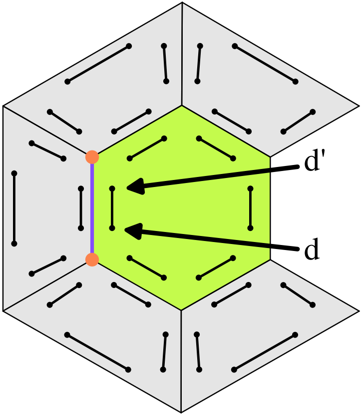
Color legend: 0, 1, 2.

Gmaps built as graphs.

d identifies the orange vertex, the purple edge, and the green face.

[1] Damiand et al. 2014.

Generalized maps[1]



Color legend: 0, 1, 2.

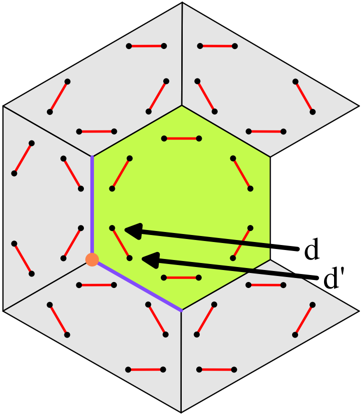
Gmaps built as graphs.

0-arc:

- distinct vertices.
- same edge and faces.

[1] Damiand et al. 2014.

Generalized maps[1]



Color legend: 0, 1, 2.

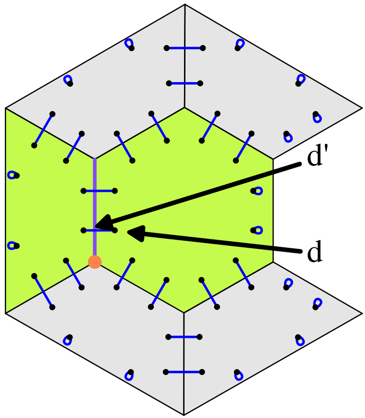
Gmaps built as graphs.

1-arc:

- distinct edges.
- same vertices and faces.

[1] Damiand et al. 2014.

Generalized maps[1]



Color legend: 0, 1, 2.

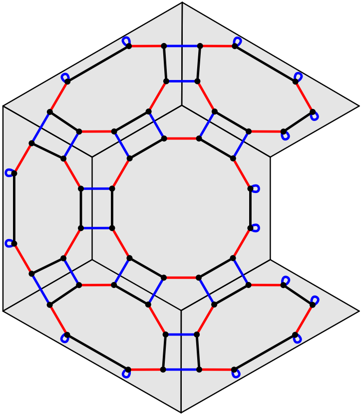
Gmaps built as graphs.

2-arc:

- distinct faces.
- same vertices and edges.

[1] Damiand et al. 2014.

Generalized maps[1]



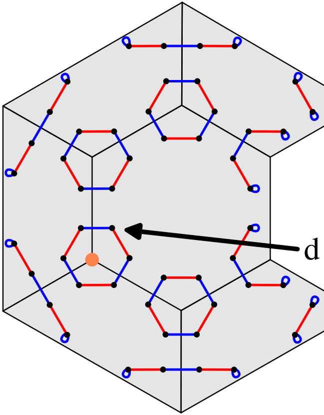
Color legend: 0, 1, 2.

Gmaps built as graphs.

Full topology obtained by superimposing the relations.

[1] Damiand et al. 2014.

Orbits and topological cells

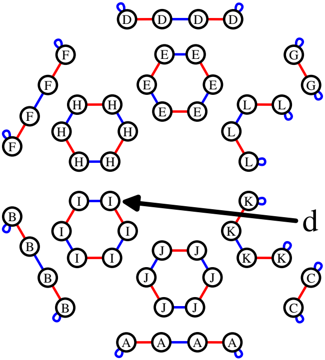


Color legend: 0, 1, 2.

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

- Vertices: orbits $\langle 1, 2 \rangle$.

Orbits and topological cells

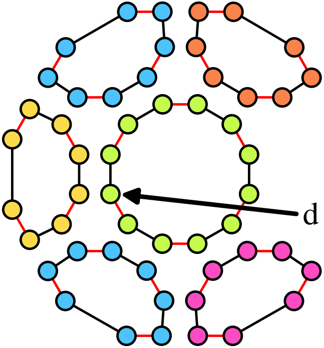


Color legend: 0, 1, 2.

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

- Vertices: orbits $\langle 1, 2 \rangle$.
  - ▶ Carry positions.

Orbits and topological cells



Color legend: 0, 1, 2.

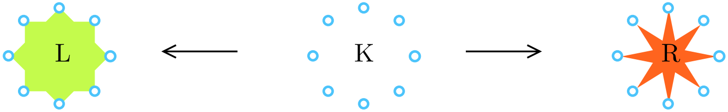
▶ Orbit (encode topological cell):
Graph induced by a subset $\langle o \rangle \subseteq [\![0, n]\!]$ of dimensions.

- Vertices: orbits $\langle 1, 2 \rangle$.
- Faces: orbits $\langle 0, 1 \rangle$.
  - ▶ Carry colors.

Formalizing modeling operations

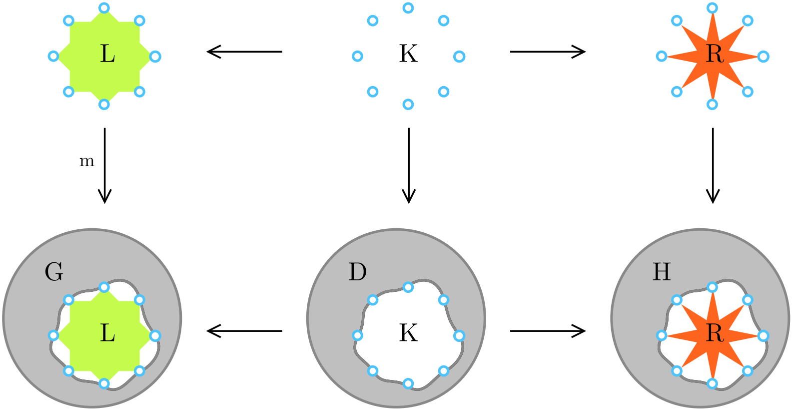
- Operations on Gmaps are designed as graph rewriting rules.

Graph transformation rules[1]



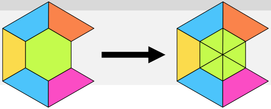
[1] Rozenberg 1997; Ehrig et al. 2006; Heckel et al. 2020.

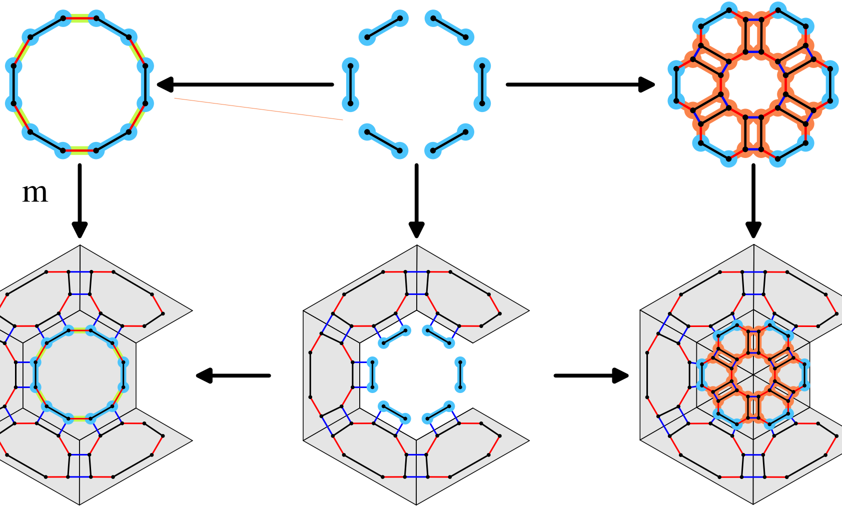
Graph transformation rules[1]



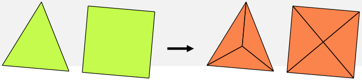
[1] Rozenberg 1997; Ehrig et al. 2006; Heckel et al. 2020.

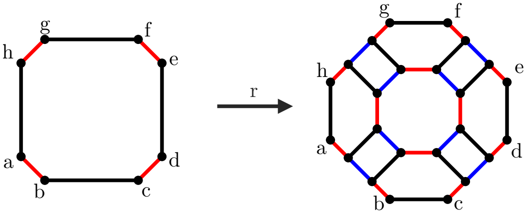
Rewriting Gmaps



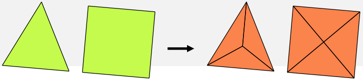


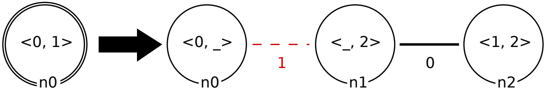
Orbit rewriting

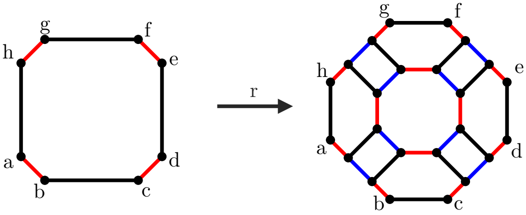




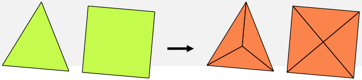
Orbit rewriting

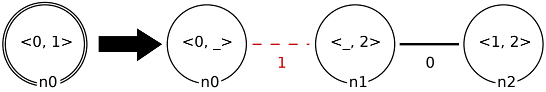


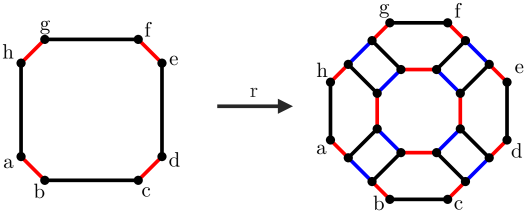




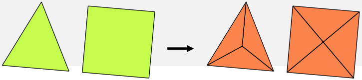
Orbit rewriting

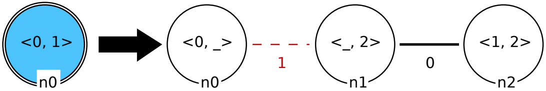


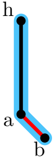


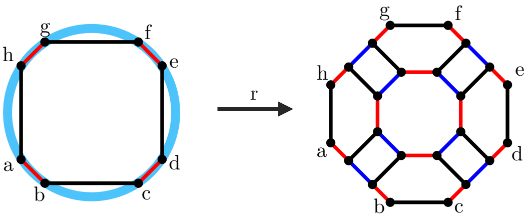


Orbit rewriting

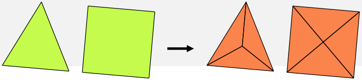


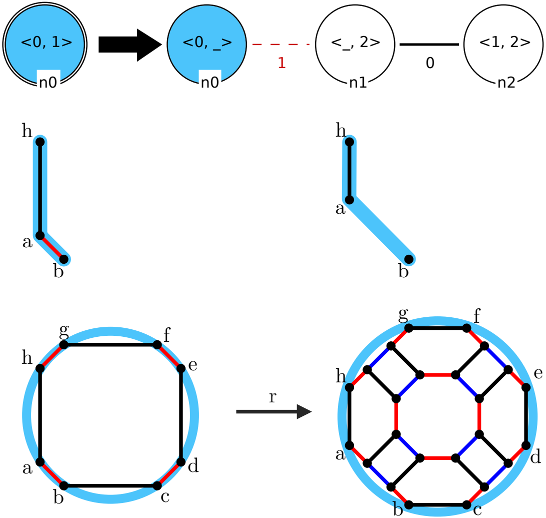




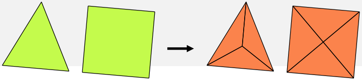


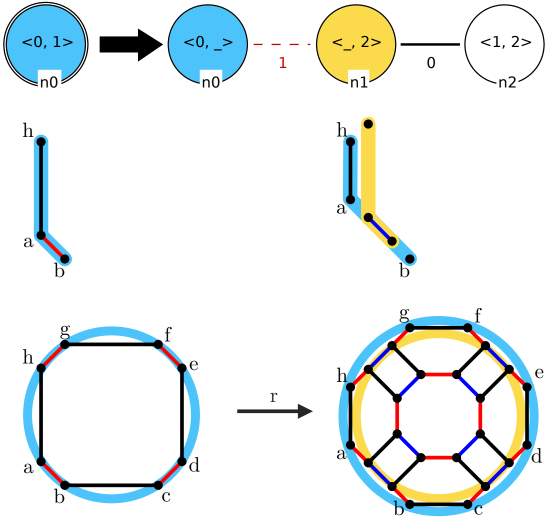
Orbit rewriting



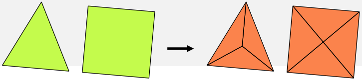


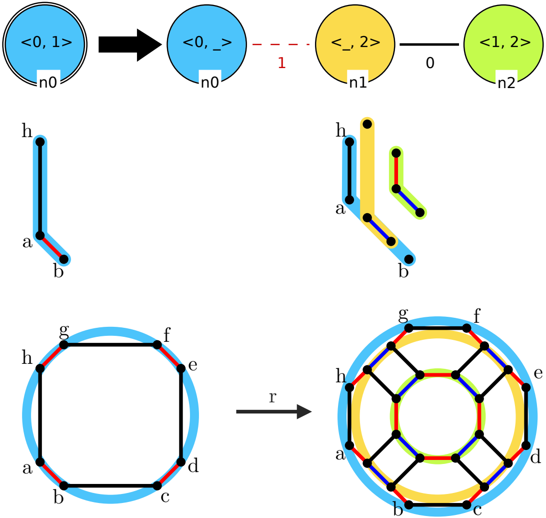
Orbit rewriting



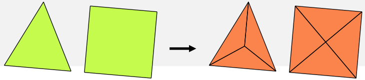


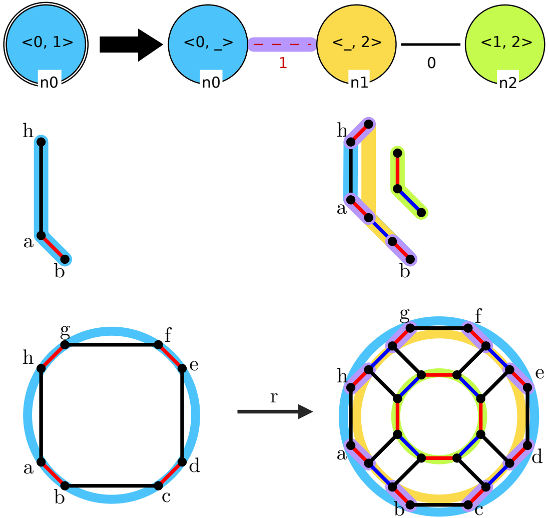
Orbit rewriting



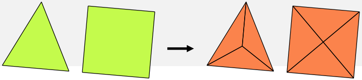


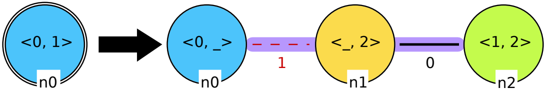
Orbit rewriting

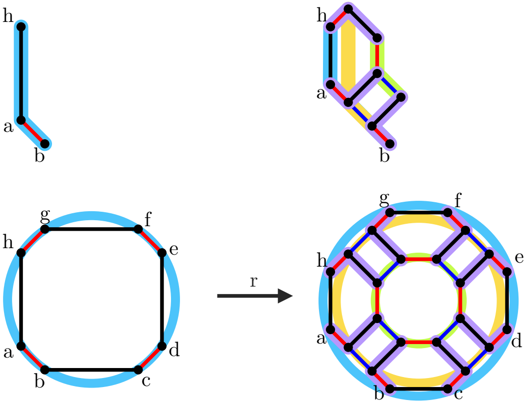




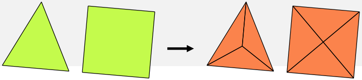
Orbit rewriting

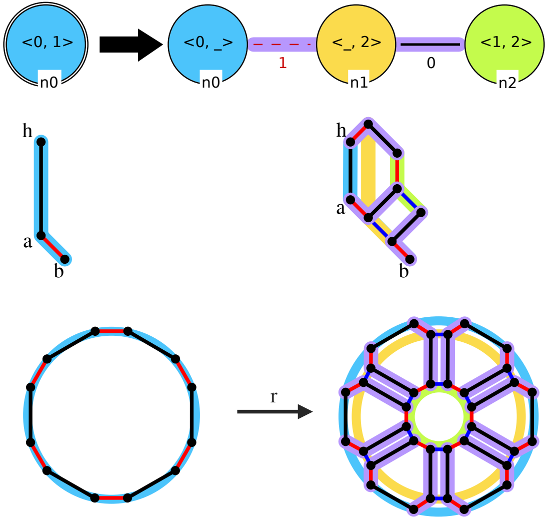






Orbit rewriting

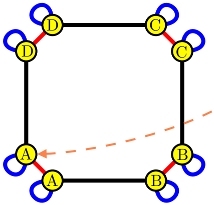


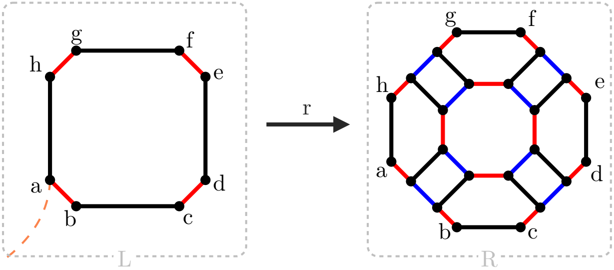


Modifying geometric values[1]

[1] Bellet et al. 2017.

Modifying geometric values[1]



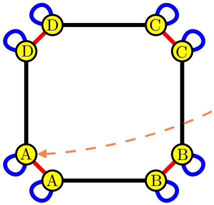


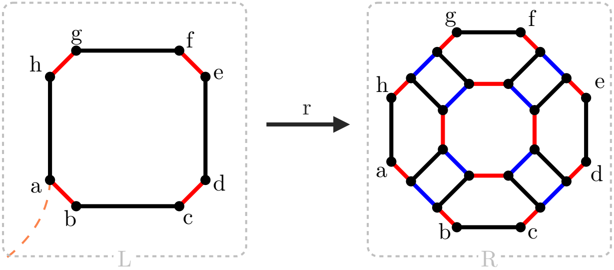
Embedding expressions modeled with algebraic data types:

- add
- middle
- scale
- ...

[1] Bellet et al. 2017.

Modifying geometric values[1]



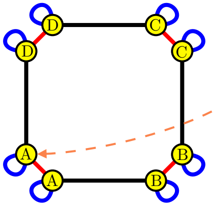


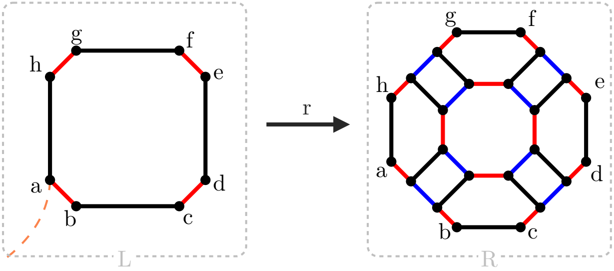
Embedding expressions are extended with topological operators:

- 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]



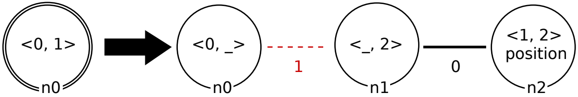


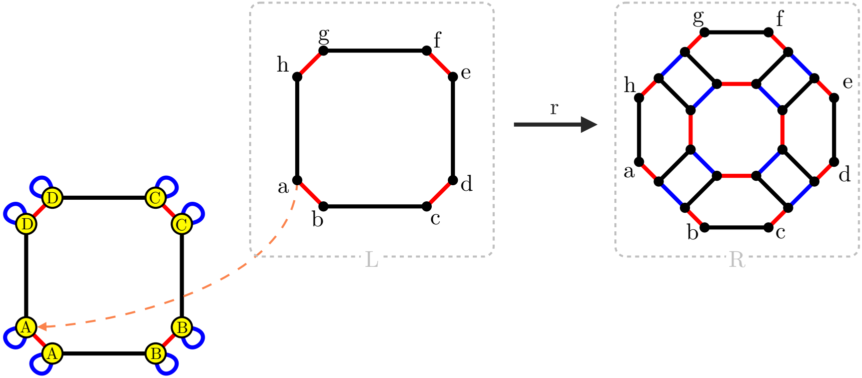
Embedding expressions are extended with topological operators:

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

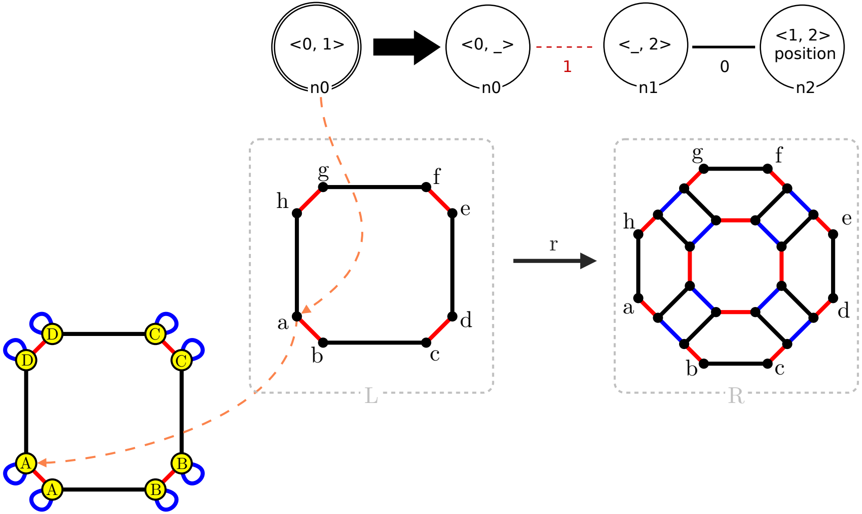
[1]Bellet et al. 2017.

Extension to schemes

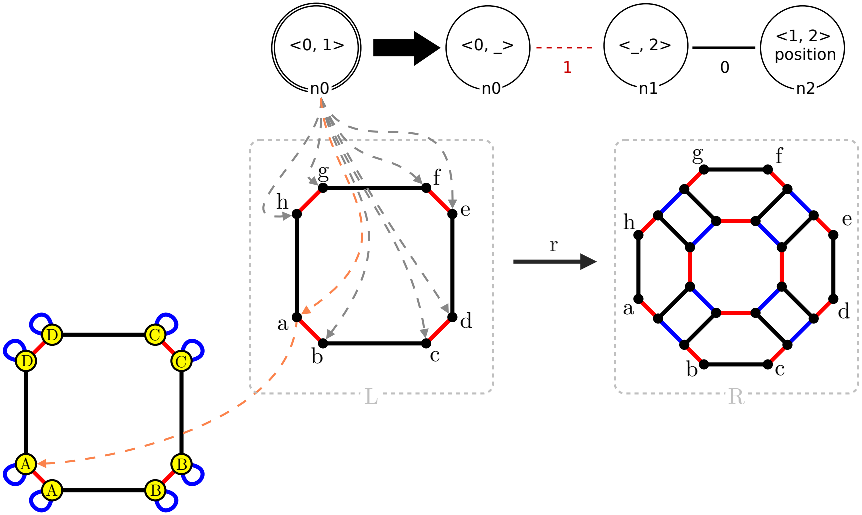




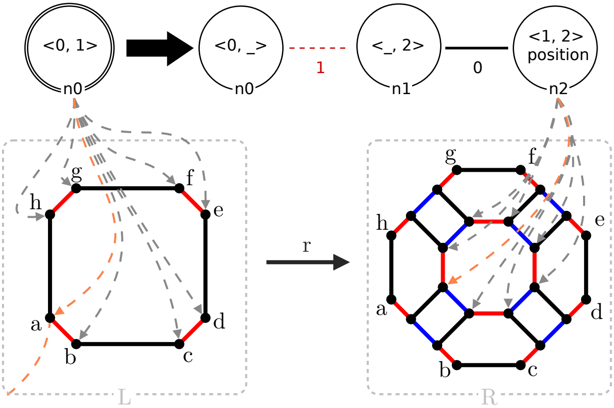
Extension to schemes

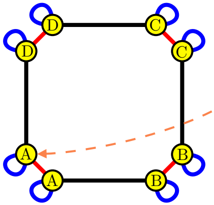


Extension to schemes



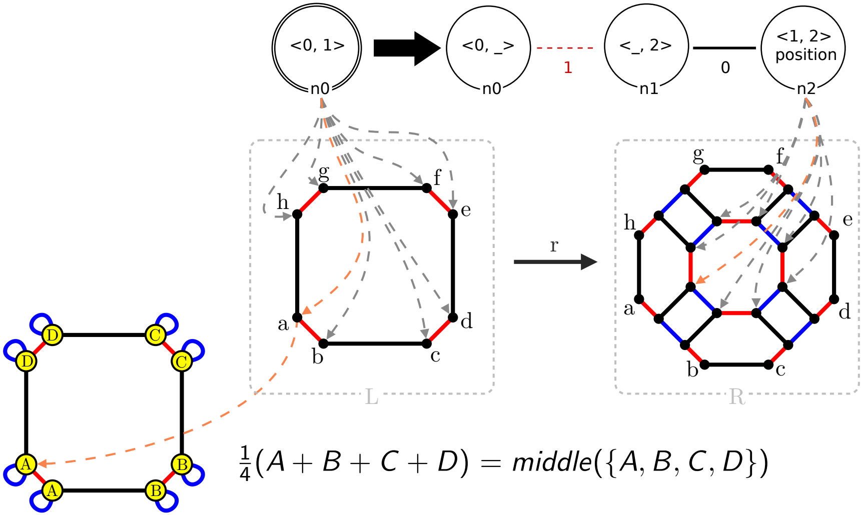
Extension to schemes





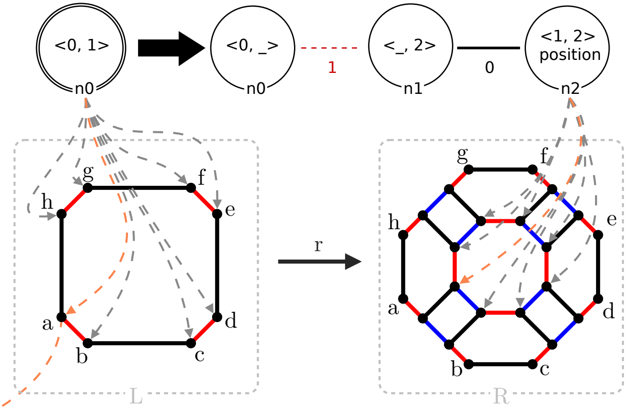
$$\frac{1}{4}(A + B + C + D)$$

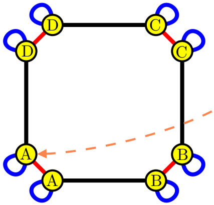
Extension to schemes



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

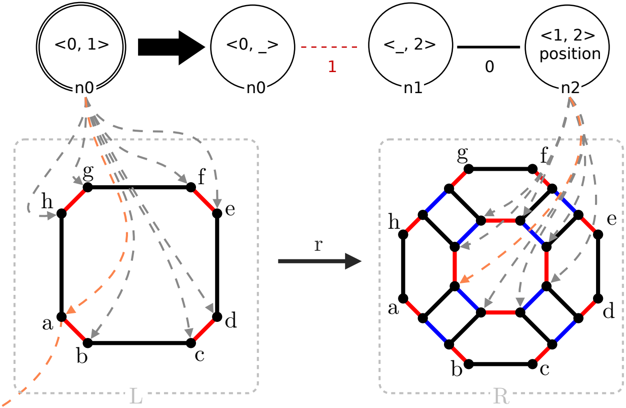
Extension to schemes

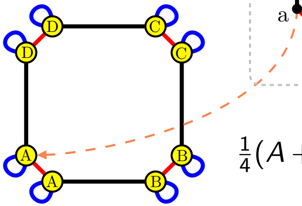




$$\begin{aligned}\frac{1}{4}(A + B + C + D) &= middle(\{A, B, C, D\})\\ &= middle(position_{\langle 0,1\rangle}(a))\end{aligned}$$

Extension to schemes

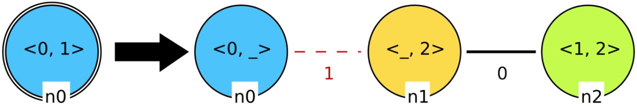




$$\begin{aligned}\frac{1}{4}(A+B+C+D) &= middle(\{A,B,C,D\}) \\ &= middle(position_{\langle 0,{\color{red}1}\rangle}(a)) \\ &= middle(position_{\langle 0,{\color{red}1}\rangle}(n0))\end{aligned}$$

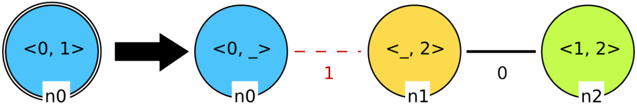
Key points

We describe geometric modeling operations with rule schemes.



Key points

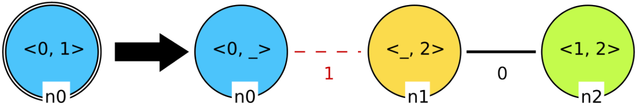
We describe geometric modeling operations with rule schemes.



- Operations described in a domain-specific language

Key points

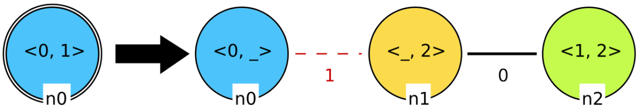
We describe geometric modeling operations with rule schemes.



- Operations described in a domain-specific language
- Each node in the rule encodes for multiple darts in the Gmaps.

Key points

We describe geometric modeling operations with rule schemes.

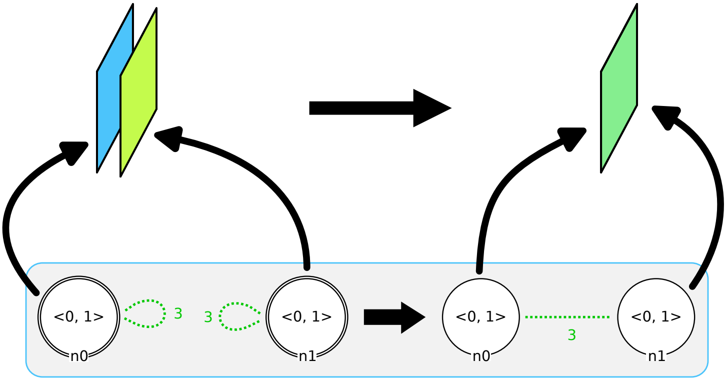


- Operations described in a domain-specific language
- Each node in the rule encodes for multiple darts in the Gmaps.
- Node names substituted by dart names during the instantiation.

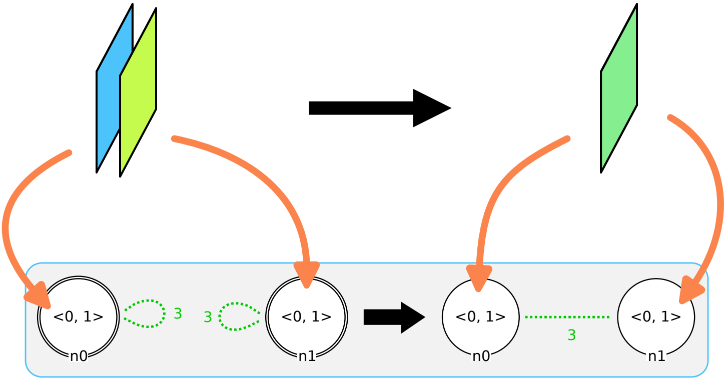
Inferring geometric modeling operations

- Retrieving the operation described by an example.

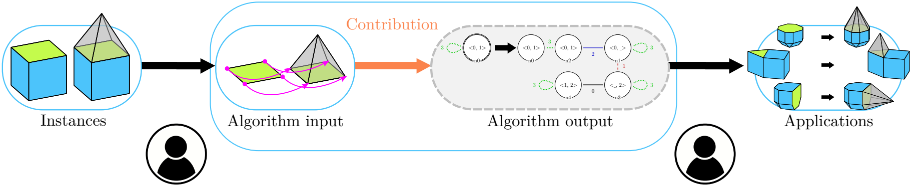
Reversing the instantiation process



Reversing the instantiation process



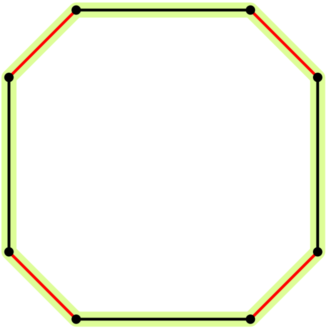
Inference workflow



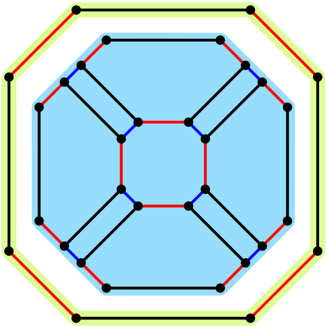
- Input: A graph G encoding the preservation relation between two partial Gmaps, an orbit type $\langle o \rangle$ and a dart a of G.

- Output: A graph $\mathcal{S}$ that encodes the Jerboa rule with the variable $\langle o \rangle$, given that the operation is applied to the dart a.

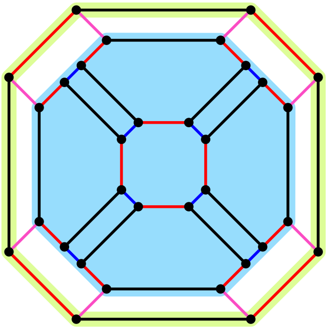
Folding a joint representation of the rule



Folding a joint representation of the rule

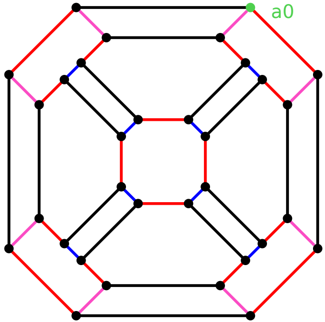


Folding a joint representation of the rule



Color legend: 0, 1, 2, κ.

Folding a joint representation of the rule



Color legend: 0, 1, 2, κ.

Besides the two Gmaps and the preservation links, we chose a dart in the initial Gmap and an orbit type.

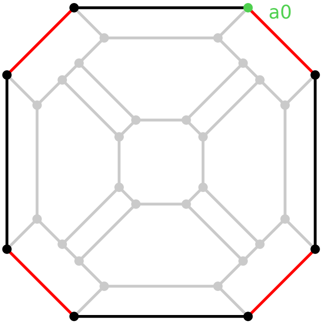
- Graph traversal algorithm.

Iteratively applying two foldings:

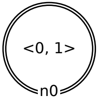
- Folding of a node.
- Folding of the arcs.

- Illustration on face triangulation with the orbit type $\langle 0, 1 \rangle$ and the dart $a0$.

Execution

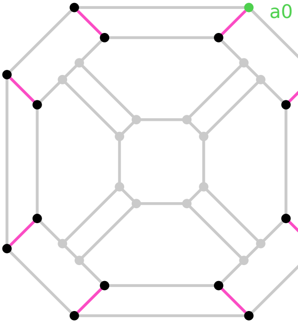


Creation of the hook (orbit $\langle 0, 1 \rangle$).

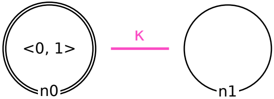


Color legend: 0, 1, 2, κ.

Execution

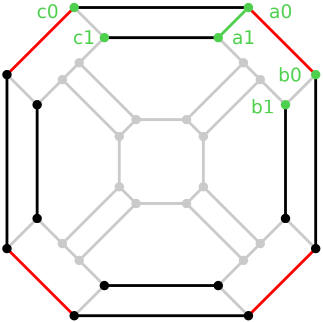


Folding of the arcs.



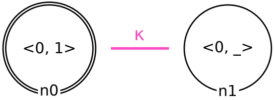
Color legend: 0, 1, 2, κ.

Execution

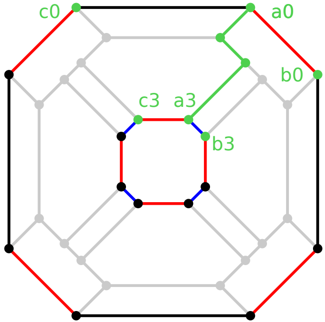


Color legend: 0, 1, 2, κ.

Folding of a node.

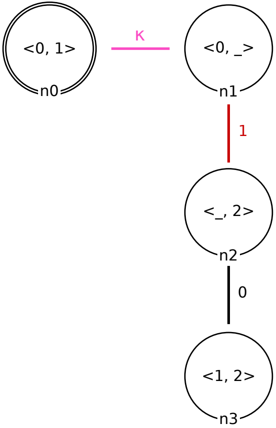


Execution

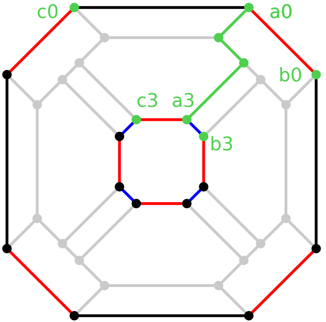


Color legend: 0, 1, 2, κ.

The algorithm terminates.

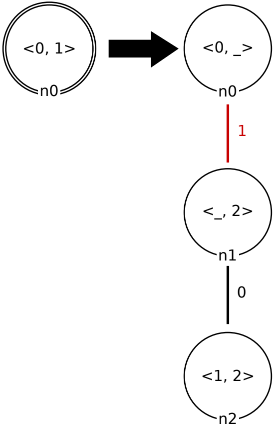


Execution



Color legend: 0, 1, 2, κ.

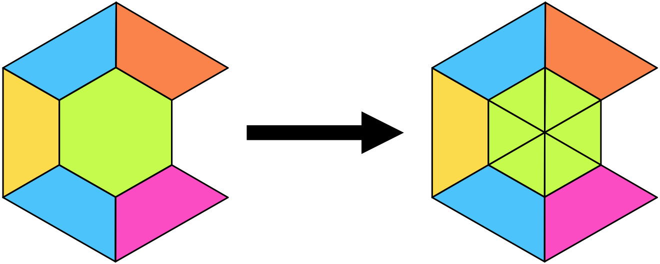
Splitting the joint representation.



Results[1]

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

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

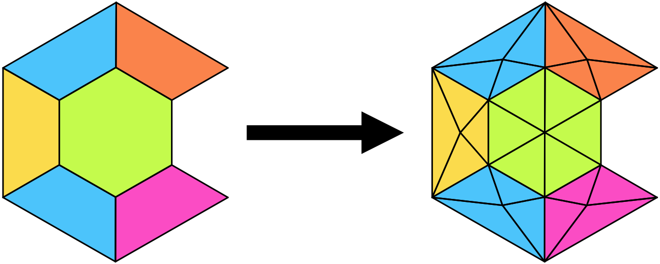


[1] Pascual et al. 2022a.

Results[1]

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

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

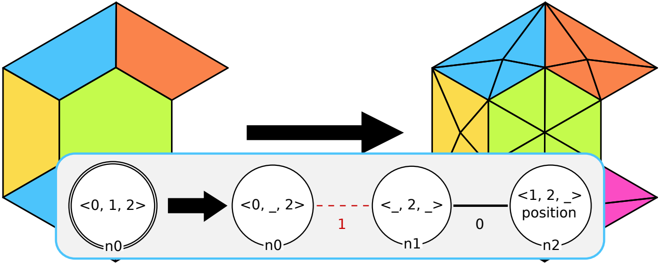


[1]Pascual et al. 2022a.

Results[1]

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

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



[1] Pascual et al. 2022a.

Method for the geometric modifications (positions)

Method for the geometric modifications (positions)

- Hypothesis: The vertex positions of the target object C are obtained as affine combinations of vertex positions in the initial object O.

Method for the geometric modifications (positions)

▶ Hypothesis: The vertex positions of the target object C are obtained as affine combinations of vertex positions in the initial object O.

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 for the geometric modifications (positions)

▶ Hypothesis: The vertex positions of the target object C are obtained as affine combinations of vertex positions in the initial object O.

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 for the geometric modifications (positions)

▶ Hypothesis: The vertex positions of the target object C are obtained as affine combinations of vertex positions in the initial object O.

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 for the geometric modifications (positions)

▶ Hypothesis: The vertex positions of the target object C are obtained as affine combinations of vertex positions in the initial object O.

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

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

where:

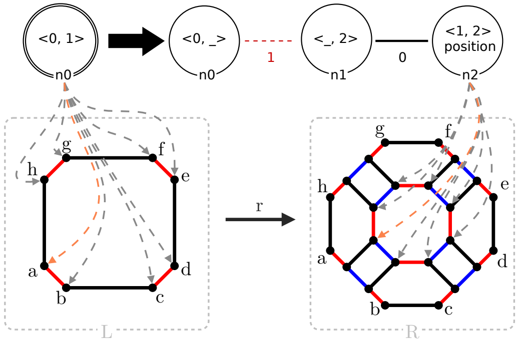
- p : target position (known)
- p_i: position of the initial vertex i (known)
- w_i: weight (unknown)
- t : translation (unknown)

Need for abstraction on schemes

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

Need for abstraction on schemes

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

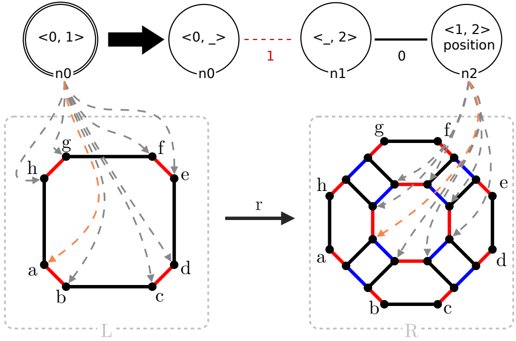


Issue: darts in the Gmap will share the same expression.

- Because rule schemes abstract topological cells.

Need for abstraction on schemes

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



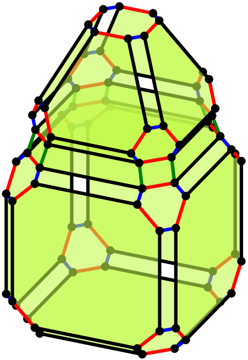
Issue: darts in the Gmap will share the same expression.

- Because rule schemes abstract topological cells.

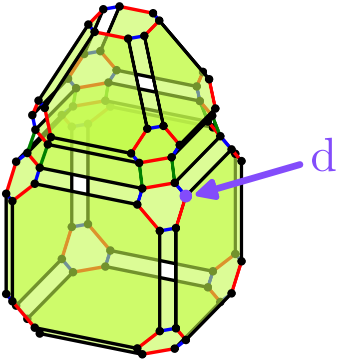
Solution: Exploit the topology.

- Use points of interest that share the same expression.

Points of interest



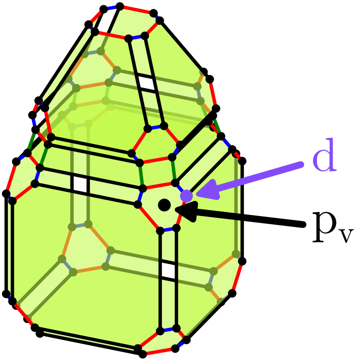
Points of interest



Points of interest

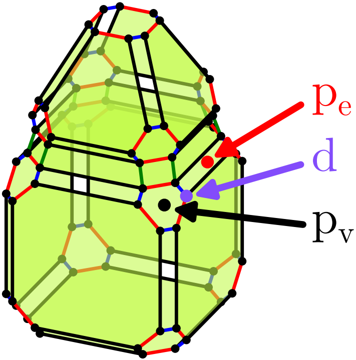
with

- p_v : vertex



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

Points of interest



with

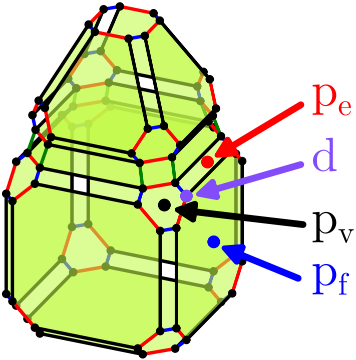
- p_v : vertex
- p_e : edge midpoint

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

Points of interest

with

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

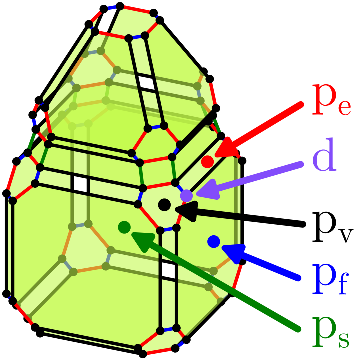


$$p_f = middle(position_{\langle 0,1,3 \rangle}(d))$$

Points of interest

with

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

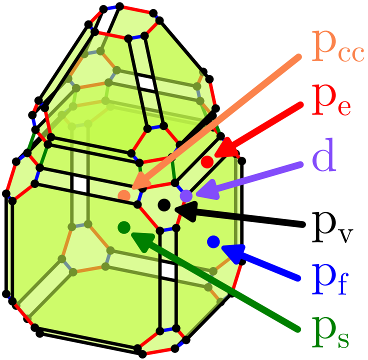


$$p_s = middle(position_{\langle 0,1,2 \rangle}(d))$$

Points of interest

with

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

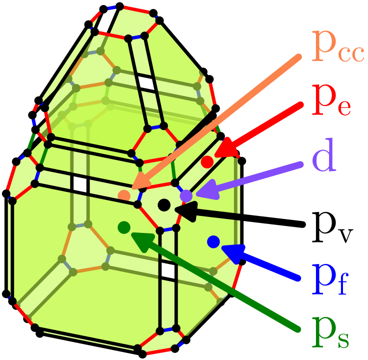


$$p_{cc} = middle(position_{\langle 0,1,2,3 \rangle}(d))$$

Points of interest

with

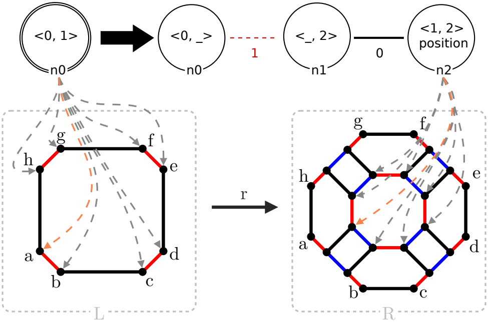
- p_v : vertex
- p_e : edge midpoint
- p_f : face barycenter
- p_s : volume barycenter
- p_{cc} : CC barycenter



Thanks to the points of interest, the system is rewritten as:

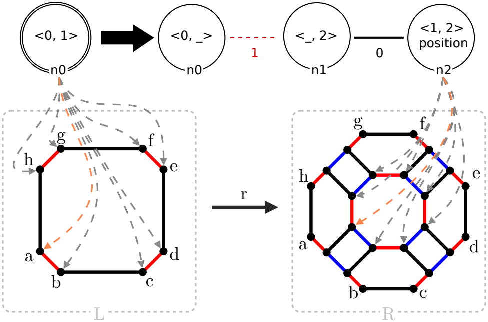
$$p = w_v p_v + w_e p_e + w_f p_f + w_s p_s + w_{cc} p_{cc} + t$$

Illustration



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

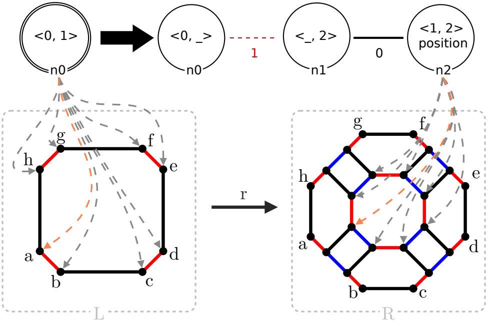
Illustration



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

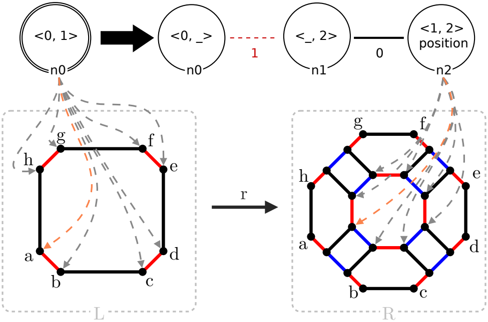


The position expression of $n2$ only depends on $n0$.

- 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

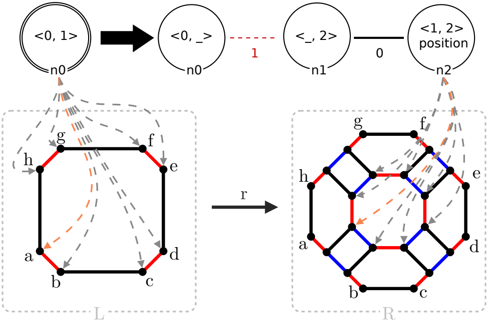


The position expression of $n2$ only depends on $n0$.

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

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

Illustration

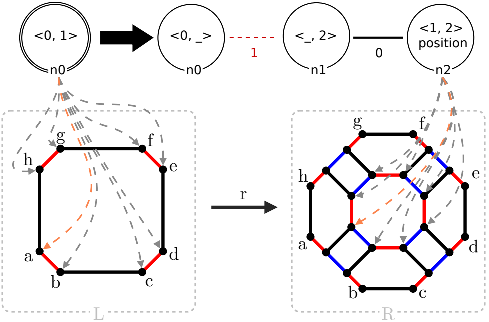


The position expression of $n2$ only depends on $n0$.

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

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

Illustration



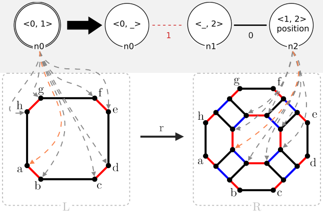
The position expression of $n2$ only depends on $n0$.

- 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)

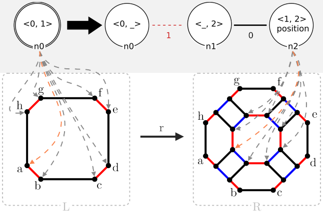
Solving the barycentric triangulation



- 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



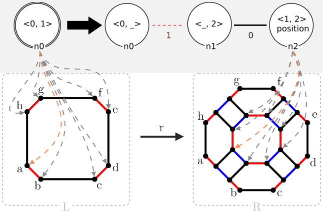
- Global equation:

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

- Generated system (only on x and y)

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

Solving the barycentric triangulation



- Global equation:

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

- Generated system (only on x and y)

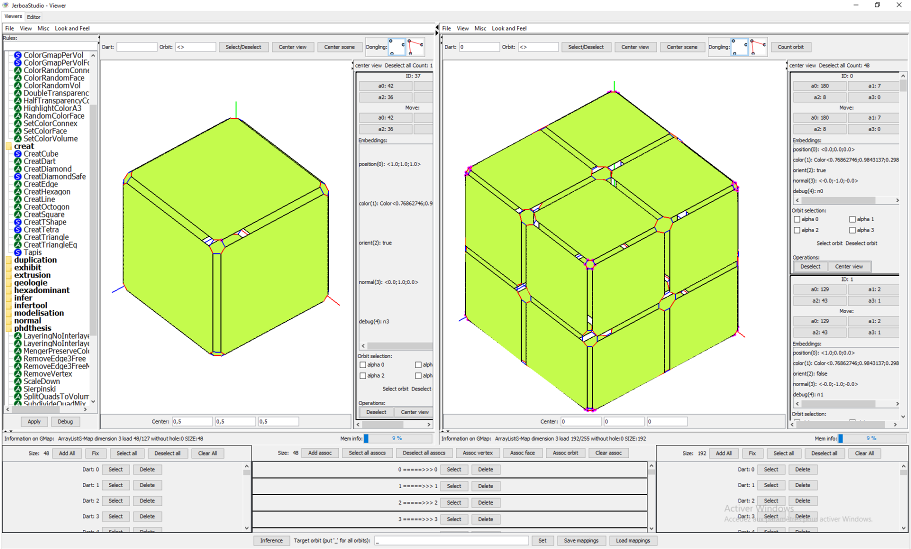
$$\begin{cases} (0.5;0.5) = w_v * (0;0) + w_e * \color{red}{(0.5;0)} + w_f * \color{blue}{(0.5;0.5)} + w_s * \color{green}{(0.5;0.5)} + w_{cc} * \color{orange}{(0.5;0.5)} + (tx;ty) \\ (0.5;0.5) = w_v * (1;0) + w_e * \color{red}{(0.5;0)} + w_f * \color{blue}{(0.5;0.5)} + w_s * \color{green}{(0.5;0.5)}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

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

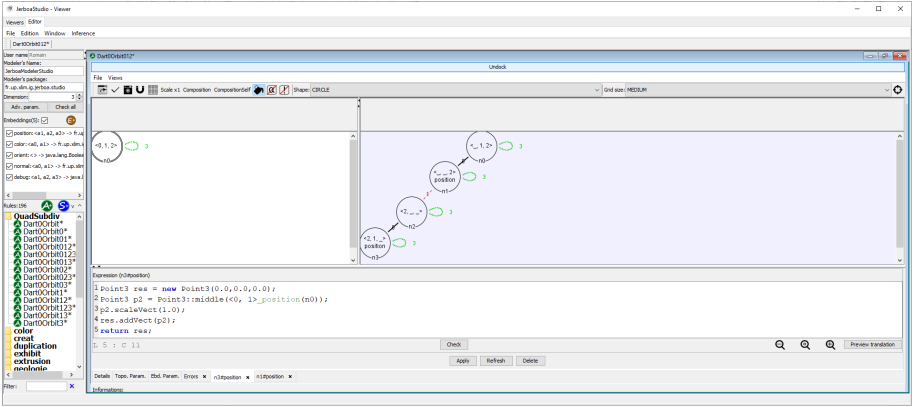
JerboaStudio and applications

- Implementation of the inference mechanism in Jerboa.

JerboaStudio: inferring the quad subdivision



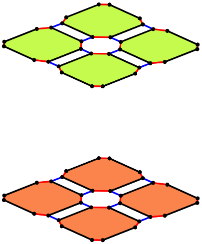
Folding the quad subdivision



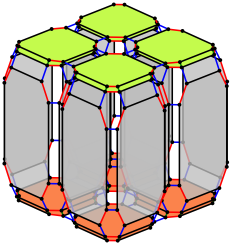
- 768 possible schemes
- 48 schemes tried (marking).
- 14 schemes built (removal of isomorphic rules).

Example inspired from geology

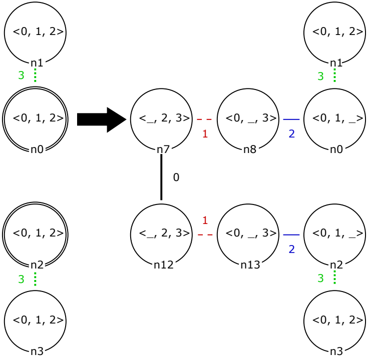
Before



After



Operation



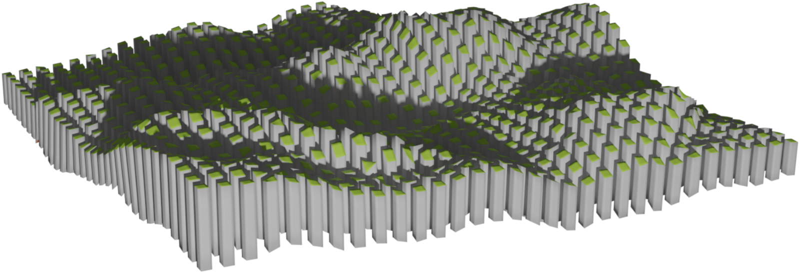
Inference time: ~ 3 ms

Example inspired from geology

Before

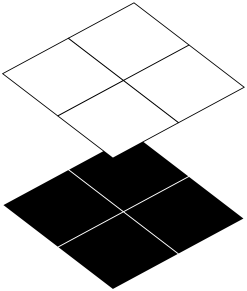
Example inspired from geology

After

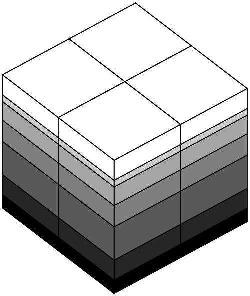


Example inspired from geology (part 2)

Before



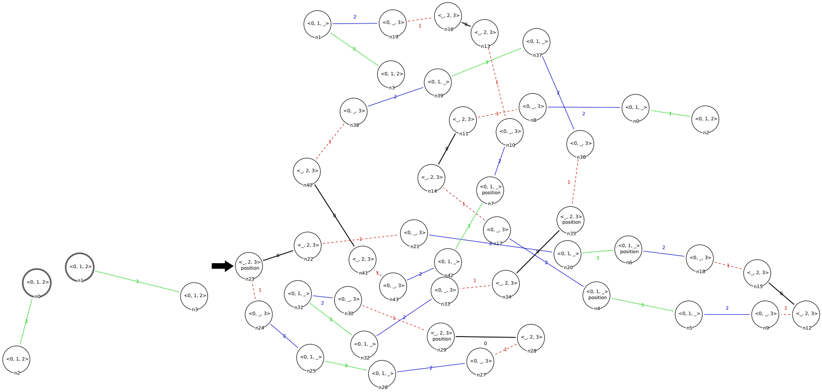
After



- We infer interpolations both for the positions and the colors.

Example inspired from geology (part 2)

Operation



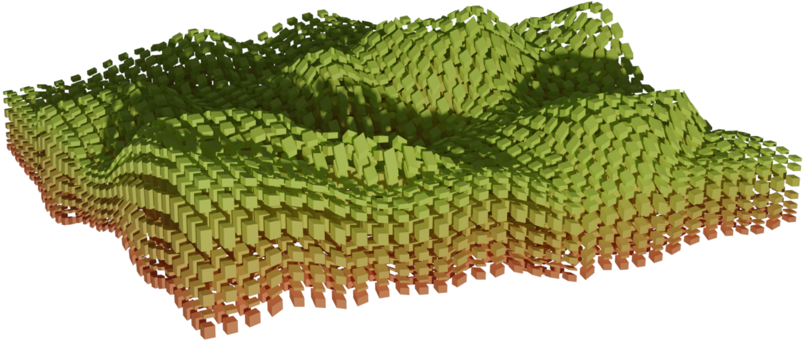
Inference time: $\sim$ 26 ms for the topology,
$\sim$ 549 ms for the embedding expressions

Example inspired from geology (part 2)

Before

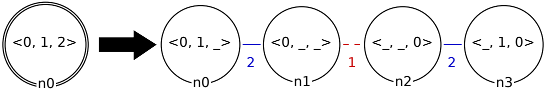
Example inspired from geology (part 2)

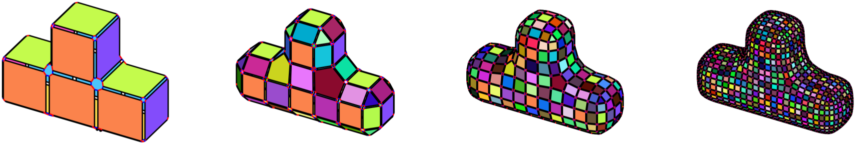
After



Doo-Sabin subdivision[1]

- Rule scheme used and inferred:

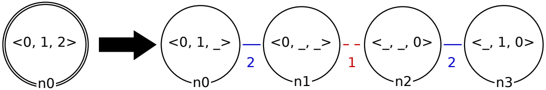


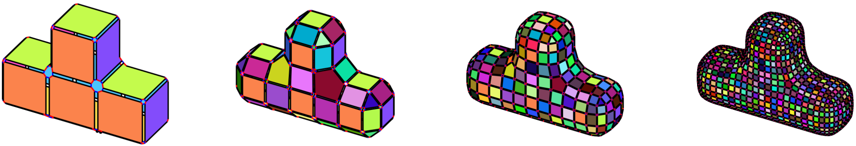


[1]Doo et al. 1978.

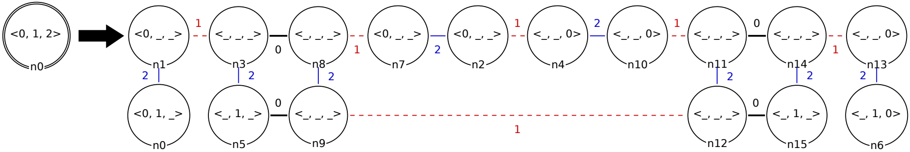
Doo-Sabin subdivision[1]

- Rule scheme used and inferred:





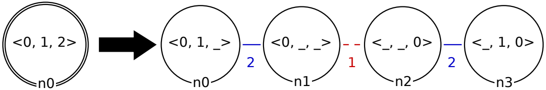
- *2nd* iteration:

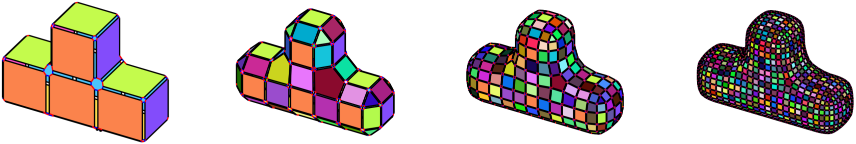


[1] Doo et al. 1978.

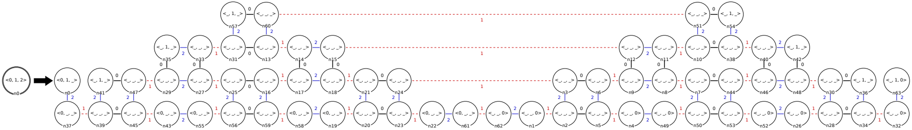
Doo-Sabin subdivision[1]

- Rule scheme used and inferred:



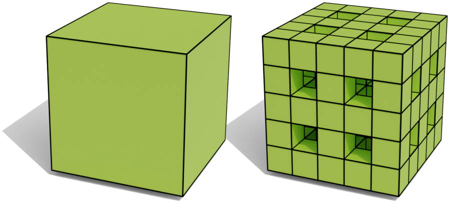


- 3rd iteration:



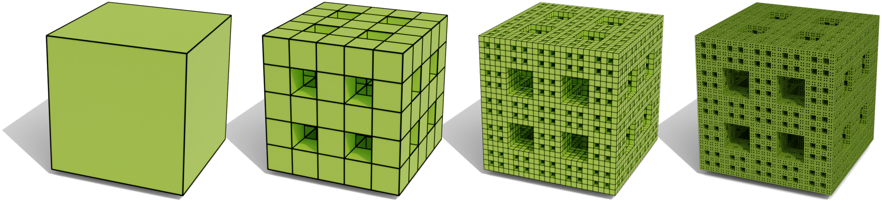
[1]Doo et al. 1978.

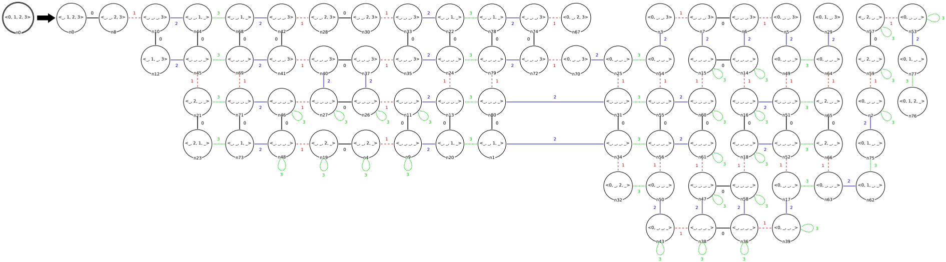
Menger $(2, 2, 2)$[1]



[1]Richaume et al. 2019.

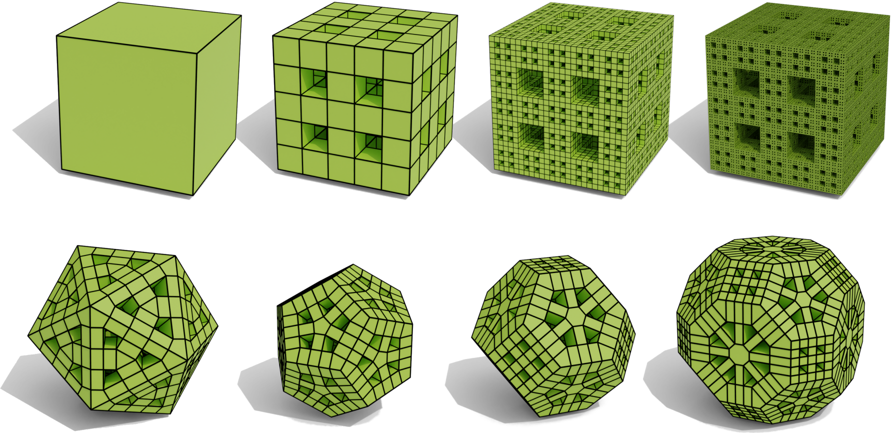
Menger $(2, 2, 2)$[1]





[1] Richaume et al. 2019.

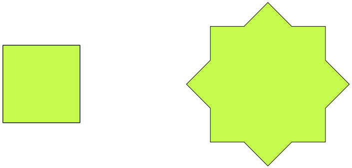
Menger $(2, 2, 2)$[1]



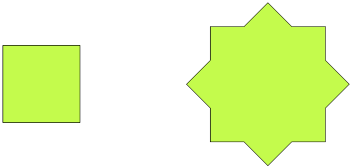
[1]Richaume et al. 2019.

Edge cases

- Von Koch's snowflake generated with L-systems

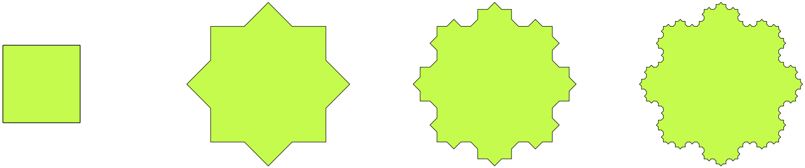


- Inferred:

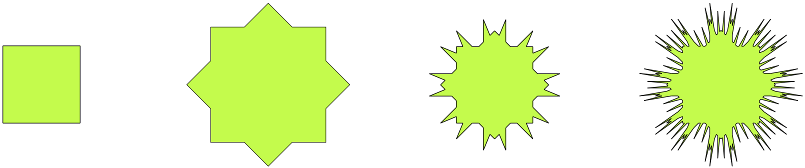


Edge cases

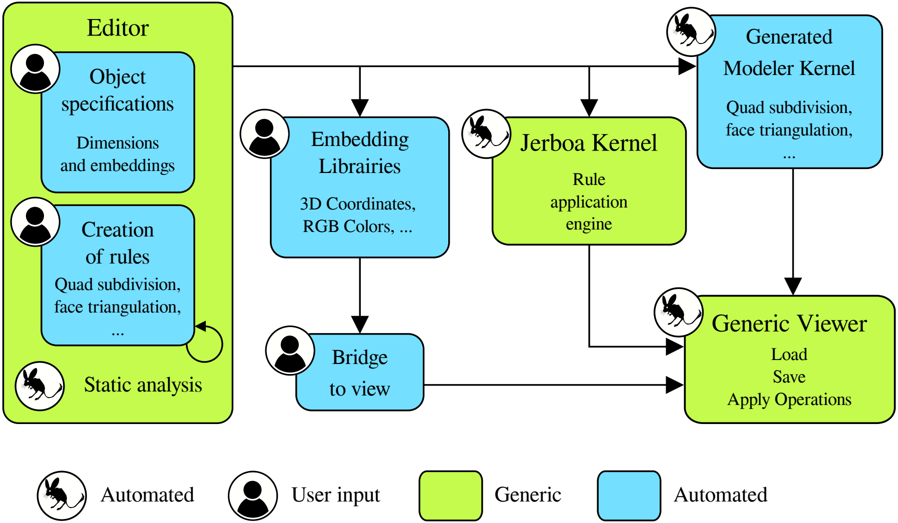
- Von Koch's snowflake generated with L-systems



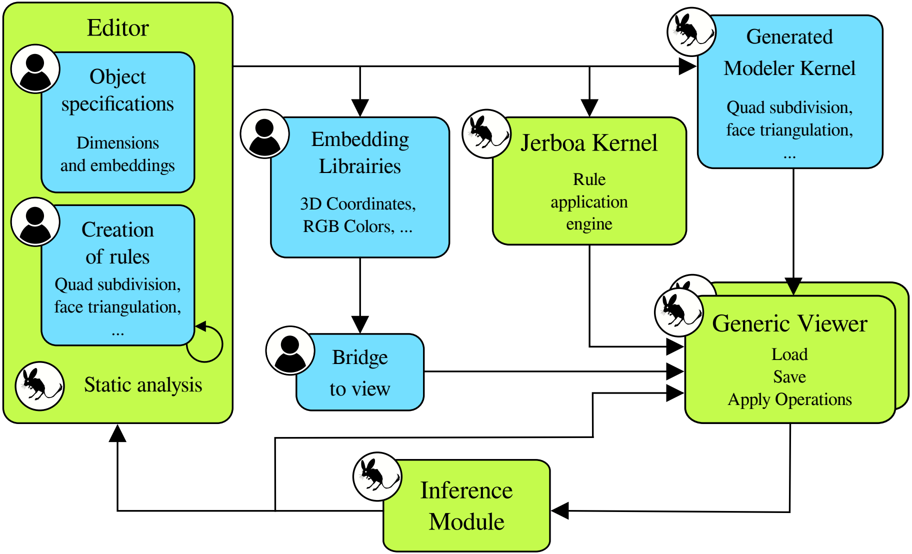
- Inferred:



JerboaStudio's architecture



JerboaStudio's architecture

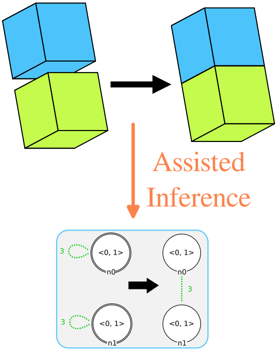


Conclusion

- Current research topics.

Main contributions

- Inference of modeling operations:
  - Topological folding algorithm[1]
  - Values of interest and CSP
- JerboaStudio.

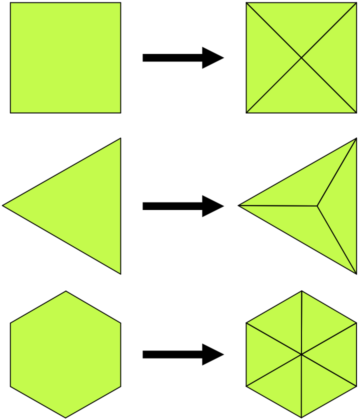


- Graph transformations for geometric modeling:
  - Graph products[2]
  - Rule completion[3]
- Unified framework for combinatorial maps.

[1] Pascual et al. 2022a
[2] Pascual et al. 2022b
[3] Arnould et al. 2022

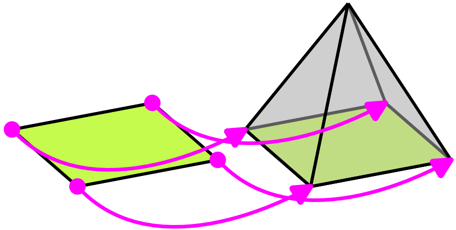
Future works

- Instance generation
  - Create objects on which a given rule scheme is applicable.



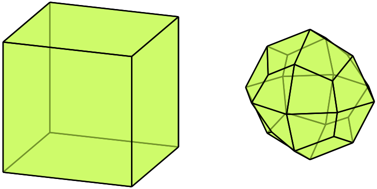
Future works

- Instance generation
- Automatic mapping
  - Cumbersome step in the inference workflow.



Future works

- Instance generation
- Automatic mapping
- Other hypotheses for the geometric inference
  - Most subdivision schemes rely on other computations: the Catmull-Clark subdivision.[1]



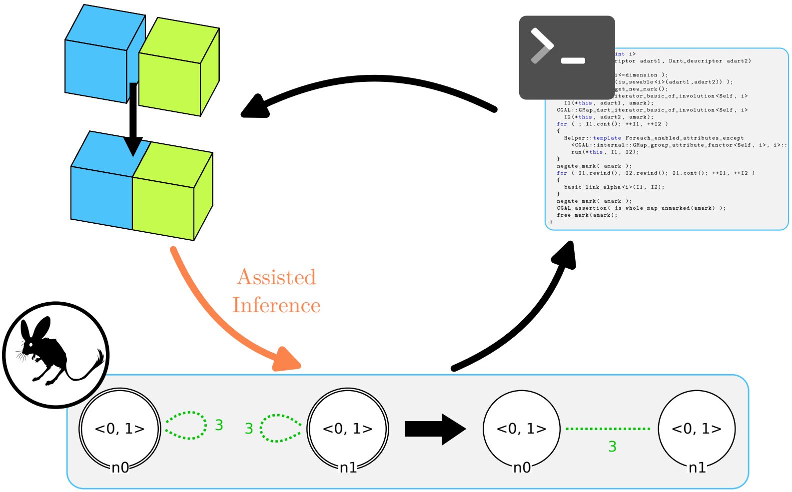
```
// From Catmull and Clark 1978, Recursively generated
// B-spline surfaces on arbitrary topological meshes

// n1#position
// midpoint of the incident face
Point3 face1Mid = Point3::middle(<0,1>_position(n0));
// midpoint of the adjacent face
Point3 face2Mid = Point3::middle(<0,1>_position(n0@2));
// average of the face points
Point3 faceMid = Point3::middle(face1Mid, face2Mid);
// midpoint of the edge
Point3 edgeMid = Point3::middle(<0>_position(n0));
// average of the edge and face points
return Point3::middle(faceMid, edgeMid);
```

Out of scope

[1] Catmull et al. 1978.

Thank you for listening



References I

Alshanqiti, Abdullah et al. (Aug. 25, 2016). "Visual contract extractor: a tool for reverse engineering visual contracts using dynamic analysis". In: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. ASE 2016. New York, NY, USA: Association for Computing Machinery, pp. 816–821. ISBN: 978-1-4503-3845-5. DOI: 10.1145/2970276.2970287.

Arnould, Agnès et al. (Oct. 18, 2022). "Preserving consistency in geometric modeling with graph transformations". In: Mathematical Structures in Computer Science. Publisher: Cambridge University Press, pp. 1–48. ISSN: 0960-1295, 1469-8072. DOI: 10.1017/S0960129522000226.

References II

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.

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.

References III

- Catmull, E. et al. (Nov. 1, 1978). "Recursively generated B-spline surfaces on arbitrary topological meshes". In: Computer-Aided Design 10.6, pp. 350–355. ISSN: 0010-4485. DOI: 10.1016/0010-4485(78)90110-0.
- 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.
- Dinella, Elizabeth et al. (2020). "Hoppity: Learning Graph Transformations to Detect and Fix Bugs in Programs". In: International Conference on Learning Representations (ICLR), p. 17.
- 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.

References IV

- 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.
- Emilien, Arnaud et al. (July 27, 2015). "WorldBrush: interactive example-based synthesis of procedural virtual worlds". In: ACM Transactions on Graphics 34.4, 106:1–106:11. ISSN: 0730-0301. DOI: 10.1145/2766975.
- Guo, Jianwei et al. (June 15, 2020). "Inverse Procedural Modeling of Branching Structures by Inferring L-Systems". In: ACM Transactions on Graphics 39.5, 155:1–155:13. ISSN: 0730-0301. DOI: 10.1145/3394105.

References V

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.

Kania, Kacper et al. (Oct. 20, 2020). "UCSG-Net – Unsupervised Discovering of Constructive Solid Geometry Tree". In: Advances in Neural Information Processing Systems 33 (NeurIPS 2020). DOI: 10.48550/arXiv.2006.09102.

Liu, Hsueh-Ti Derek et al. (July 8, 2020). "Neural subdivision". In: ACM Transactions on Graphics 39.4, 124:124:1–124:124:16. ISSN: 0730-0301. DOI: 10.1145/3386569.3392418.

References VI

López-Fernández, Jesús J. et al. (2019). "An example is worth a thousand words: Creating graphical modelling environments by example". In: Software & Systems Modeling 18.2. Publisher: Springer, pp. 961–993. DOI: 10.1007/s10270-017-0632-7.

Pascual, Romain et al. (May 14, 2022a). "Inferring topological operations on generalized maps: Application to subdivision schemes". In: Graphics and Visual Computing 6, p. 200049. ISSN: 2666-6294. DOI: 10.1016/j.gvc.2022.200049.

Pascual, Romain et al. (Feb. 1, 2022b). "Topological consistency preservation with graph transformation schemes". In: Science of Computer Programming 214, p. 102728. ISSN: 0167-6423. DOI: 10.1016/j.scico.2021.102728.

References VII

- 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.
- 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.

References VIII

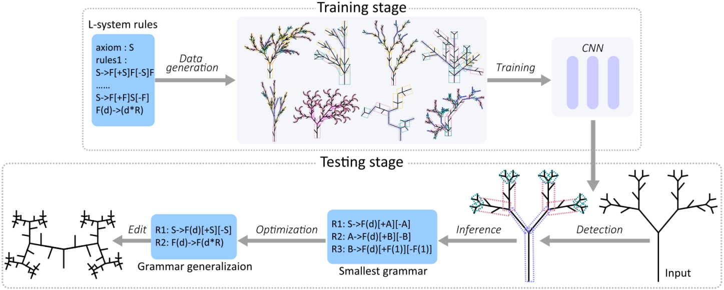
- Santos, Edmar et al. (Nov. 2009). "Obtaining L-Systems Rules from Strings". In: 2009 3rd Southern Conference on Computational Modeling, pp. 143–149. DOI: 10.1109/MCSUL.2009.21.
- Sharma, Gopal et al. (Mar. 31, 2018). "CSGNet: Neural Shape Parser for Constructive Solid Geometry". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5515–5523. DOI: 10.48550/arXiv.1712.08290. arXiv: 1712.08290.
- Šťava, Ondrej et al. (2010). "Inverse Procedural Modeling by Automatic Generation of L-systems". In: Computer Graphics Forum 29.2, pp. 665–674. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2009.01636.x.
- Wu, Fuzhang et al. (July 27, 2014). "Inverse procedural modeling of facade layouts". In: ACM Transactions on Graphics 33.4, 121:1–121:10. ISSN: 0730-0301. DOI: 10.1145/2601097.2601162.

References IX

Xu, Xianghao et al. (June 2021). "Inferring CAD Modeling Sequences Using Zone Graphs". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6062–6070. DOI: 10.48550/arXiv.2104.03900. arXiv: 2104.03900.

Other lines of research on inference

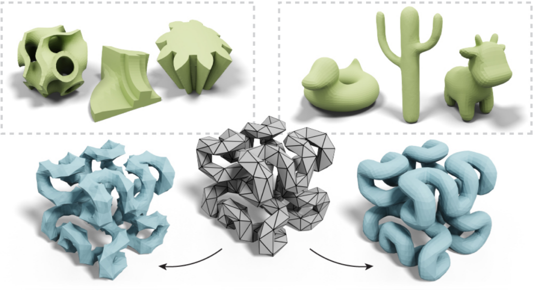
- Inferring the generation of an object:
  - Inverse procedural modeling: retrieving parameters.[1]
  - L-systems: retrieving formal rules.[2] Illustration from (Guo et al. 2020).
  - Constructive solid geometry: retrieving sequences of operations.[3]



[1]Wu et al. 2014; Emilien et al. 2015.
[2]Santos et al. 2009; Šťava et al. 2010.
[3]Sharma et al. 2018; Kania et al. 2020; Xu et al. 2021.

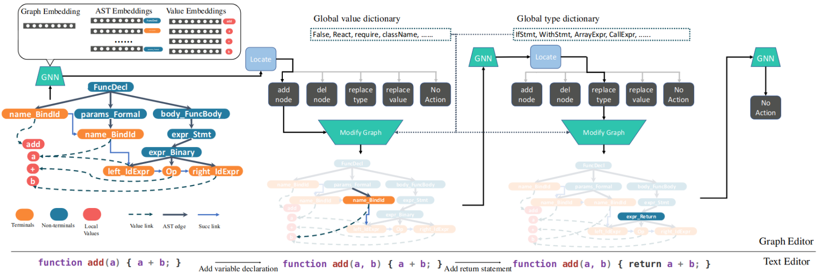
Other lines of research on inference

- Inferring the generation of an object
- Pure geometry
  - Retrieve non-linear weights of a Loop-based subdivision scheme for mesh refinement. Illustration from (Liu et al. 2020).



Other lines of research on inference

- Inferring the generation of an object
- Pure geometry
- Graph transformations
  - Domain-based inference mechanism retrieving or exploiting graph transformations.[1] Illustration from (Dinella et al. 2020).



[1]Alshanqiti et al. 2016; López-Fernández et al. 2019.

Pseudo-code for TFA

Algorithm 1: Topological folding algorithm

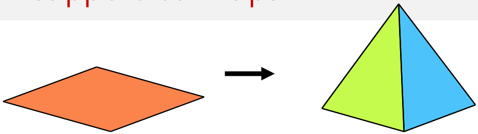
Input: A Gmap G, an orbit type $\langle o \rangle$, and a dart a of G.

Output: The scheme graph $\mathcal{S}$ such that the instantiation $\iota^{\langle o \rangle}(\mathcal{S}, G\langle o \rangle(a))$ maps (h, a) onto a.

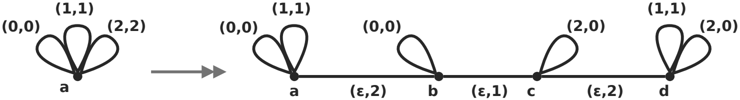
$Q \leftarrow$ empty queue
$\mathcal{S} \leftarrow \emptyset$ `// Empty scheme graph`
h $\leftarrow$ `createHook`($\mathcal{S}$,$\langle o \rangle$) `// Construction of the hook (Step 1)`
`enqueue`(Q,h)
while Q **not** empty **do** `// Traversal (Step 2)`
    $m \leftarrow$ `dequeue`(Q)
    foreach $d \in (0..n) \setminus$ `orbType`(m) **do**
        $v \leftarrow$ `arcExt`(G,m,d) `// Extension of the explicit arcs incident to v (Step 2.1)`
        `buildOrbType`(G,v) `// Construction of the orbit type decorating v (Step 2.2)`
        `enqueue`(Q,v)
return $\mathcal{S}$

Extended rules schemes with support to maps

Cone operation.

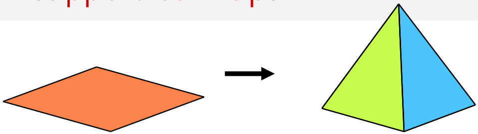


- G-map rule scheme

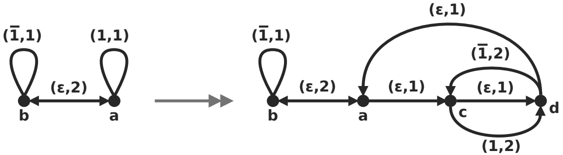


Extended rules schemes with support to maps

Cone operation.



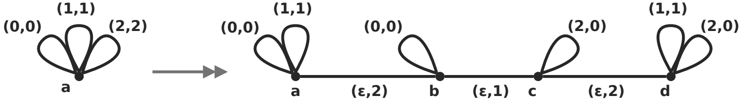
- O-map rule scheme



Extended rules schemes with support to maps

Edge rounding operation.

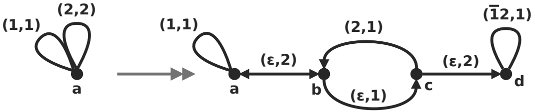
- G-map rule scheme



Extended rules schemes with support to maps

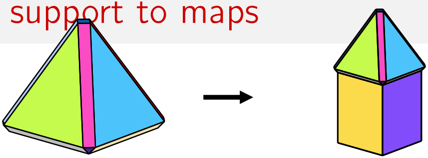
Edge rounding operation.

- O-map rule scheme

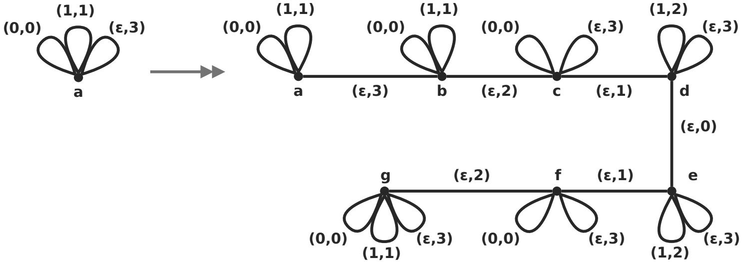


Extended rules schemes with support to maps

Face extrusion operation.

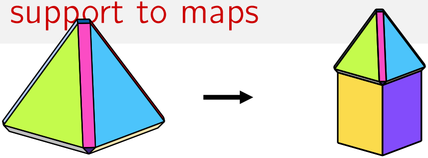


- G-map rule scheme



Extended rules schemes with support to maps

Face extrusion operation.



- O-map rule scheme