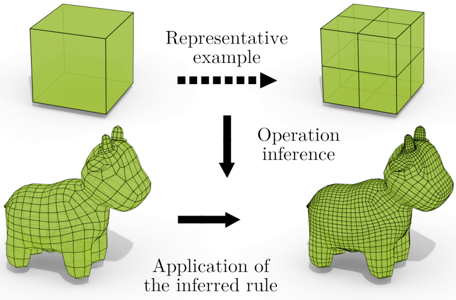
An approach for inferring geometric expressions in topology-based geometric modeling

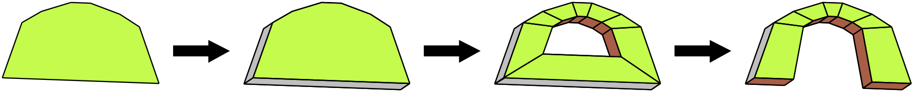
Revisited as a program synthesis problem

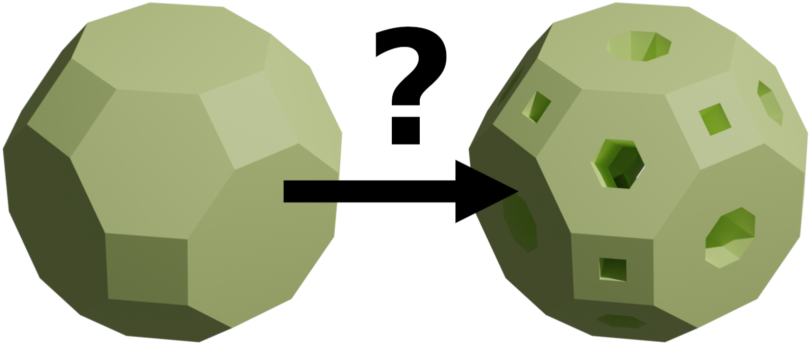


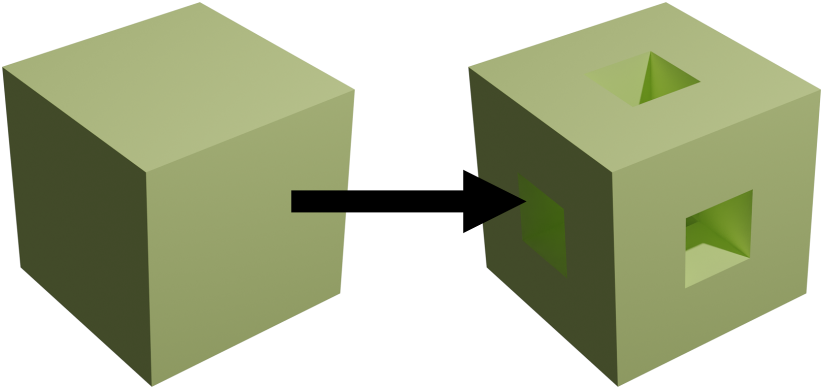
Romain Pascual

Pascale Le Gall, Hakim Belhaouari,
and Agnès Arnould

October 6, 2023
FM&AI working group at LMF



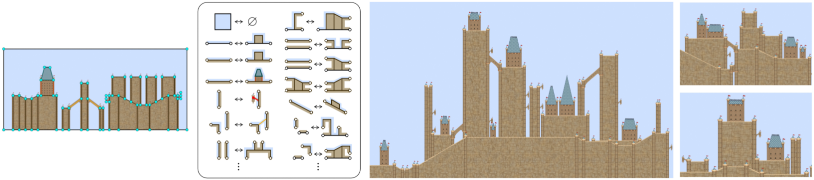




Inferring modeling operations

Inferring the generation of an object:

- Inverse procedural modeling: retrieving parameters.[1]
- L-systems: retrieving formal rules.[2]
- Constructive solid geometry: retrieving sequences of operations.[3]
- Polyhedral decomposition: retrieving a graph grammar. Illustration from (Merrell 2023)



(a) Example Shape (b) Graph Grammar (c) Generated Shapes

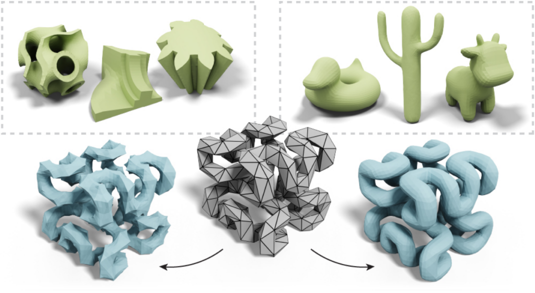
[1] Wu et al. 2014; Emilien et al. 2015.
[2] Santos et al. 2009; Šťava et al. 2010; Guo et al. 2020.
[3] Sharma et al. 2018; Kania et al. 2020; Xu et al. 2021.

Inferring modeling operations

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).



Program synthesis

How to automatically derive code from a high-level specification of the input-to-output behavior?

Program synthesis

> How to automatically derive code from a high-level specification of the input-to-output behavior?

Programming by demonstration

1. Build a theorem "for all input, there exists an output such that the specification holds."
2. Construct a proof of the theorem (proof assistant)
3. Derive a program from the proof

Program synthesis

How to automatically derive code from a high-level specification of the input-to-output behavior?

Programming by demonstration

Syntax-guided[1]

- Search-based approaches that leverage a syntactic template

[1] Alur et al. 2013.

Program synthesis

How to automatically derive code from a high-level specification of the input-to-output behavior?

Programming by demonstration

Syntax-guided[1]

(Neural approaches, LLM, etc.)

[1] Alur et al. 2013.

Example from (Alur et al. 2018)

Consider the specification

$$\forall x \forall y \;\; (x \leq f(x,y)) \;\wedge\; (y \leq f(x,y)) \;\wedge\; (f(x,y) \in \{x,y\})$$

Example from (Alur et al. 2018)

Consider the specification

$$\forall x \forall y \;\; (x \leq f(x,y)) \;\wedge\; (y \leq f(x,y)) \;\wedge\; (f(x,y) \in \{x,y\})$$

Consider the context-free grammar generated by

$$\begin{aligned} T &:= x|y|0|1|T + T|ITE(C, T, T) \\ C &:= (T \leq T)|\neg C|(C \wedge C) \end{aligned}$$

Example from (Alur et al. 2018)

Consider the specification

$$\forall x \forall y \ \ (x \leq f(x,y)) \ \wedge \ (y \leq f(x,y)) \ \wedge \ (f(x,y) \in \{x,y\})$$

Consider the context-free grammar generated by

$$\begin{aligned} T &:= x|y|0|1|T+T|ITE(C,T,T) \\ C &:= (T \leq T)|\neg C|(C \wedge C) \end{aligned}$$

Possible expression

$$f(x,y) = ITE((x \leq y), y, x)$$

Syntax-guided program synthesis

Given

- a function f, specified by a formula φ in a theory T
- a language L of admissible expressions

Find an expression $e \in L$ such that

$$\varphi[f/e] \text{ is valid modulo } T$$

Syntax-guided program synthesis

Given

- a function f, specified by a formula φ in a theory T
- a language L of admissible expressions

Find an expression $e \in L$ such that

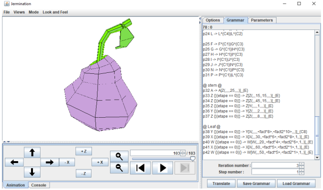
$$\varphi[f/e] \text{ is valid modulo } T$$

Programming by example[1]

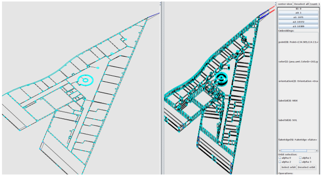
- φ derived from an input-output example
- L is a domain-specific language

[1] Gulwani et al. 2012.

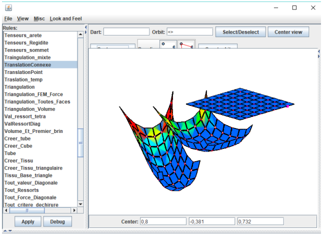
Plant growth



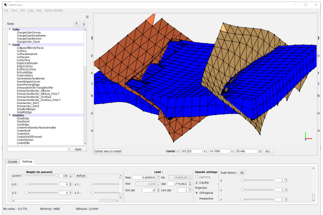
Architecture

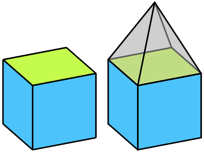


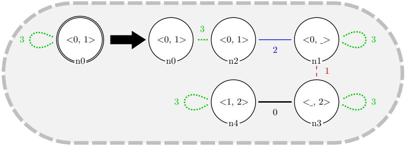
Spring-mass simulations

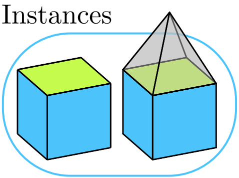


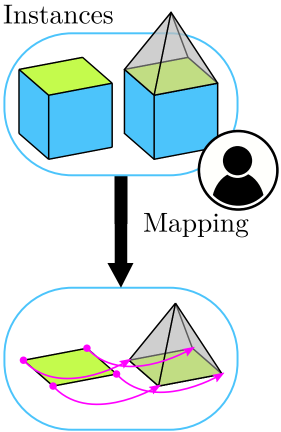
Geology

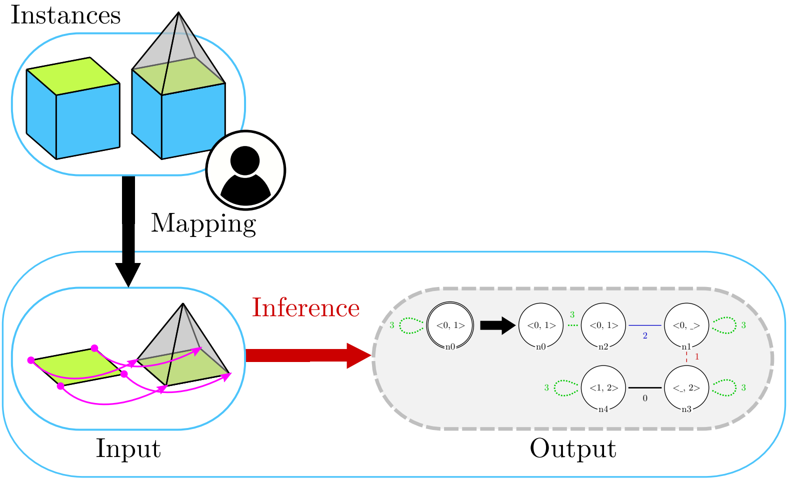


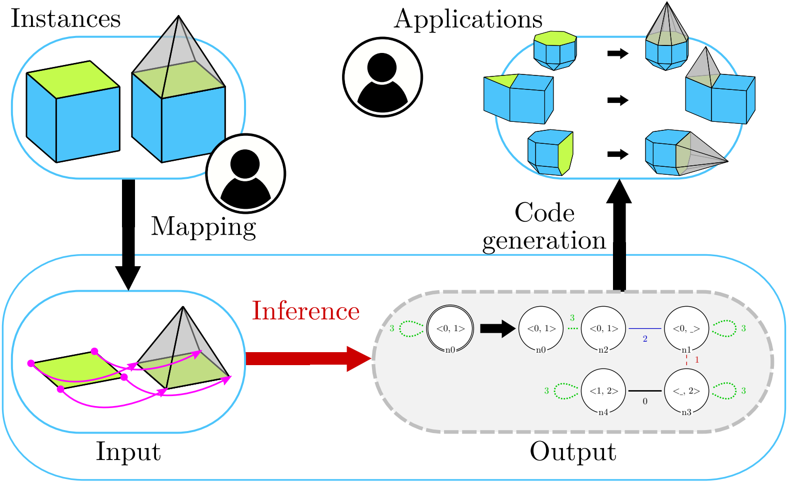



3
<0, 1>
n0
<0, 1>
n0
3
<0, 1>
n2
2
<0, _>
n1
3
1
3
<1, 2>
n4
0
<_, 2>
n3
3


Instances


Instances
Mapping

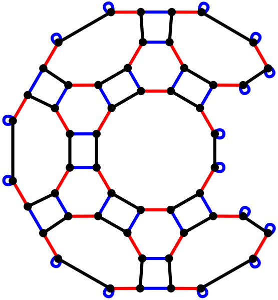

Instances
Mapping
Inference
Input
Output
<0, 1>
n0
<0, 1>
n0
<0, 1>
n2
<0, _>
n1
<1, 2>
n4
<_, 2>
n3
3
2
1
0


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

Embedded generalized maps

- How to represent objects?

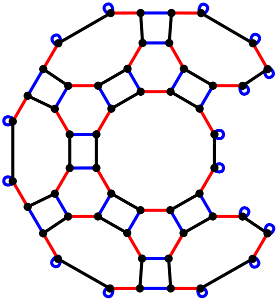
Generalized maps[1] (topology)



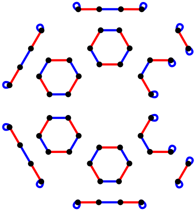
Legend: 0, 1, 2

[1] Damiand et al. 2014.

Generalized maps[1] (topology)



Orbit: Sub-graph induced by a subset $\langle o \rangle$ of dimensions



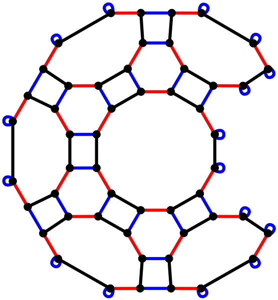
Legend: 0, 1, 2

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

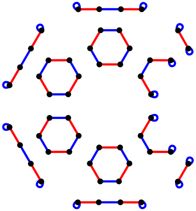
[1] Damiand et al. 2014.

Generalized maps[1] (topology)

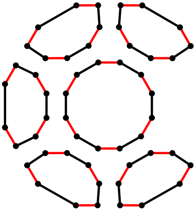
Orbit: Sub-graph induced by a subset $\langle o \rangle$ of dimensions



Legend: 0, 1, 2



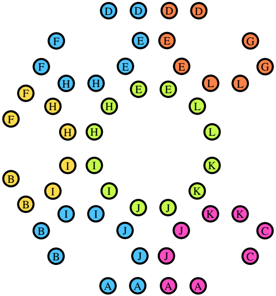
Vertices: orbits $\langle 1, 2 \rangle$



Faces: orbits $\langle 0, 1 \rangle$

[1] Damiand et al. 2014.

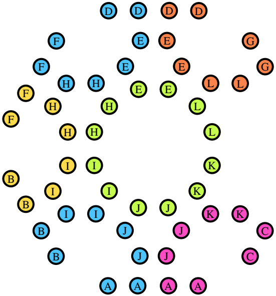
Embeddings (geometry)



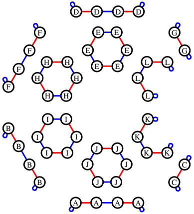
Legend: 0, 1, 2

Embeddings (geometry)

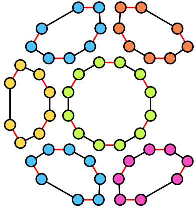
Embedding: function $\pi : \langle o_\pi \rangle \rightarrow \tau_\pi$
with τ_π an abstract data type



Legend: 0, 1, 2



$\mathit{position} : \langle 1, 2 \rangle \rightarrow \texttt{Point3}$

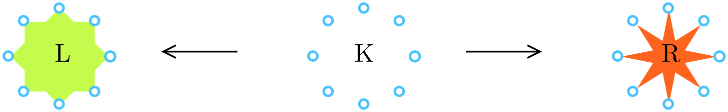


$\mathit{color} : \langle 0, 1 \rangle \rightarrow \texttt{ColorRGB}$

Graph rewriting

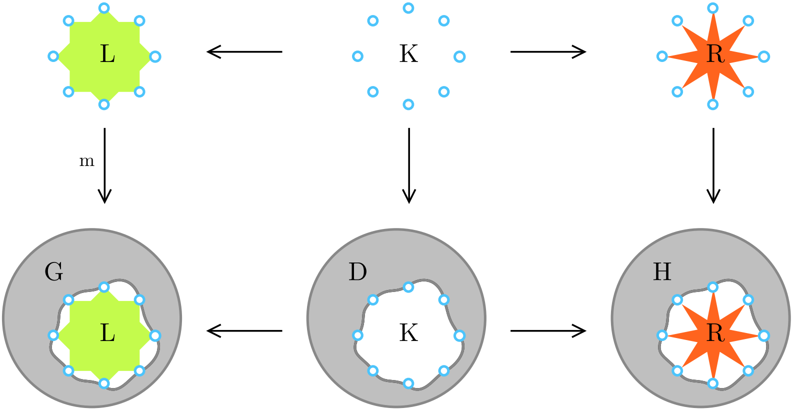
- How to formalize object transformations?

Graph transformation rules[1]



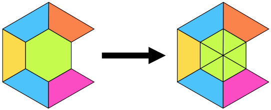
[1] Rozenberg 1997; Ehrig et al. 2006; Heckel et al. 2020.

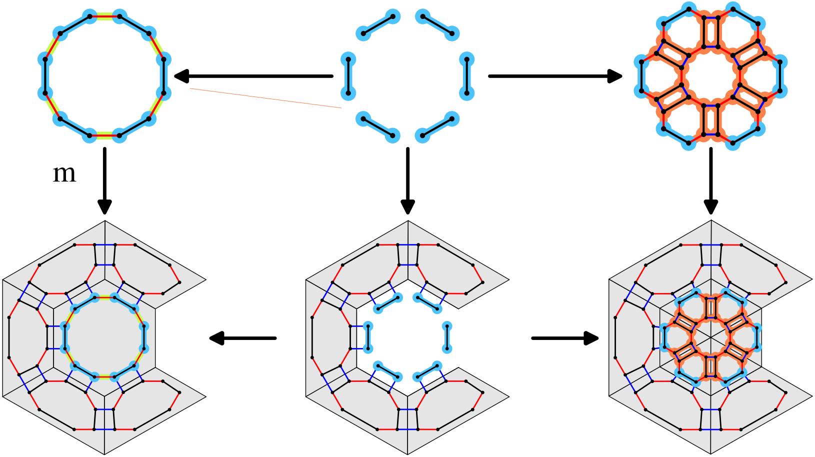
Graph transformation rules[1]



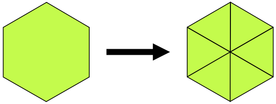
[1] Rozenberg 1997; Ehrig et al. 2006; Heckel et al. 2020.

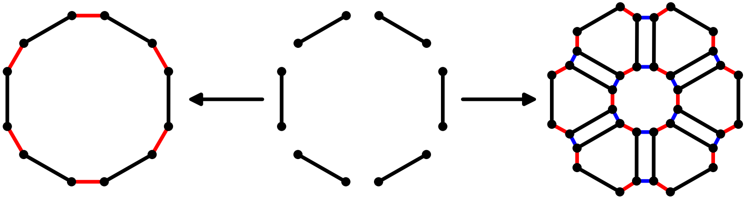
Topological rewriting of Gmaps



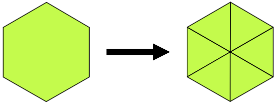


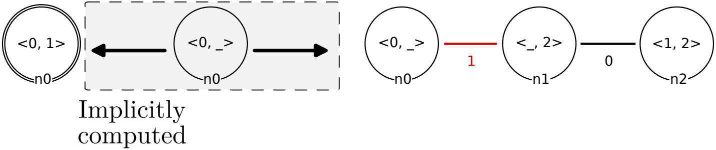
Orbit rewriting

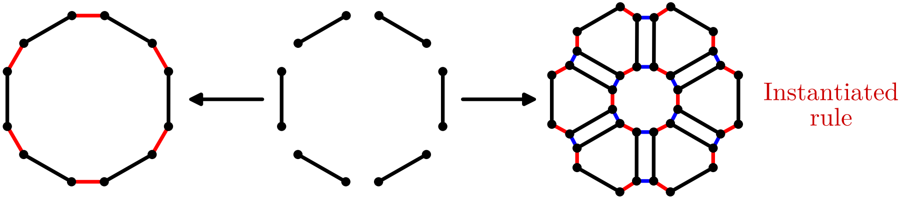




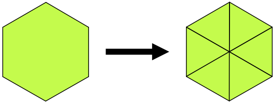
Orbit rewriting

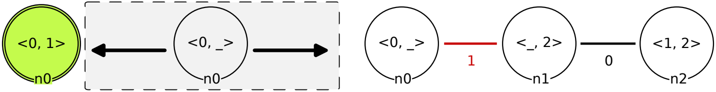






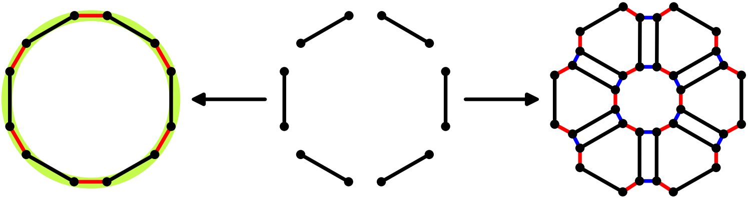
Orbit rewriting





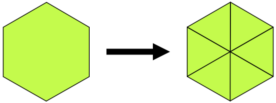
Implicitly computed

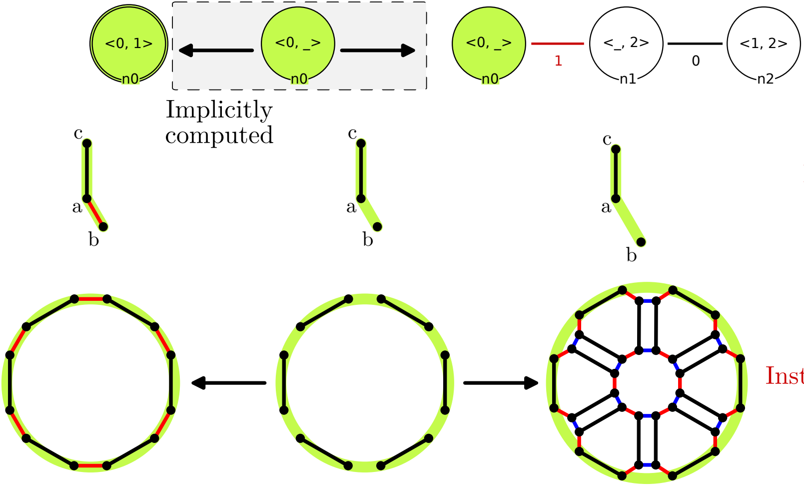
Local



Instantiated rule

Orbit rewriting

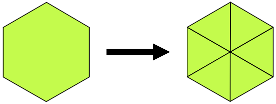


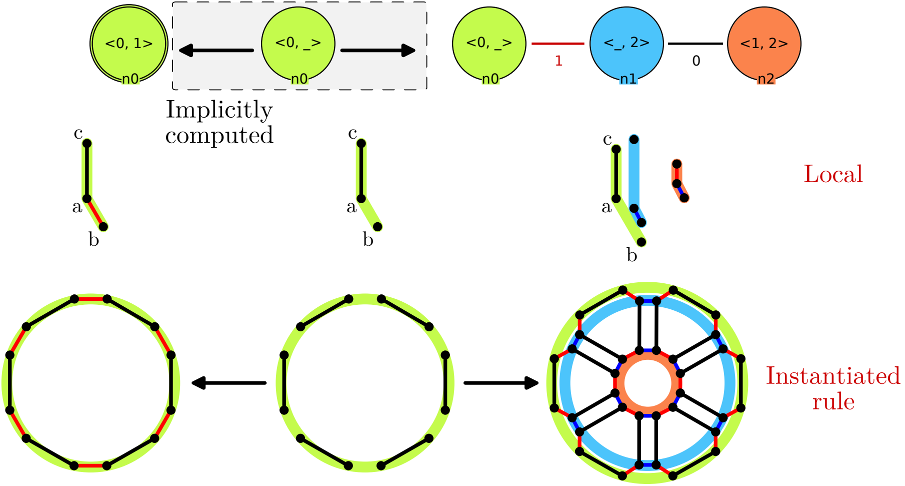


Local

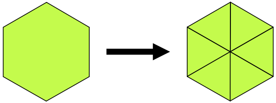
Instantiated rule

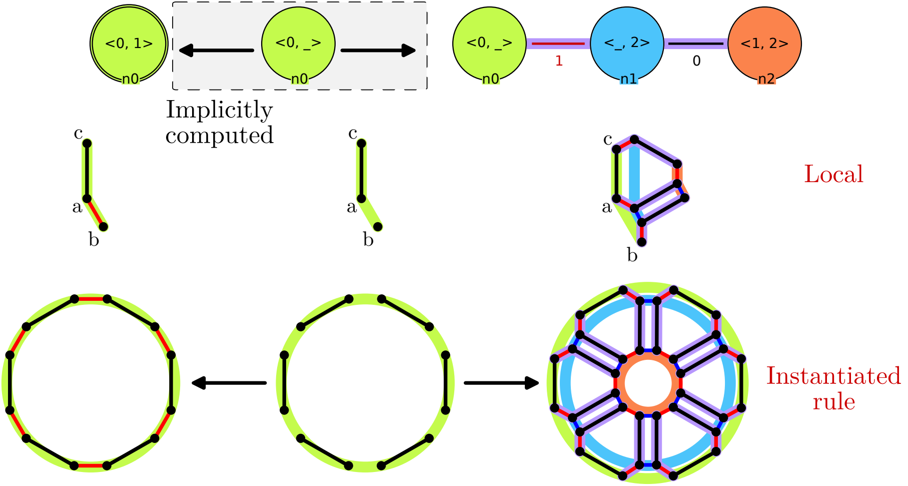
Orbit rewriting



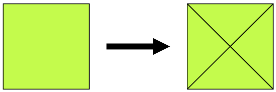


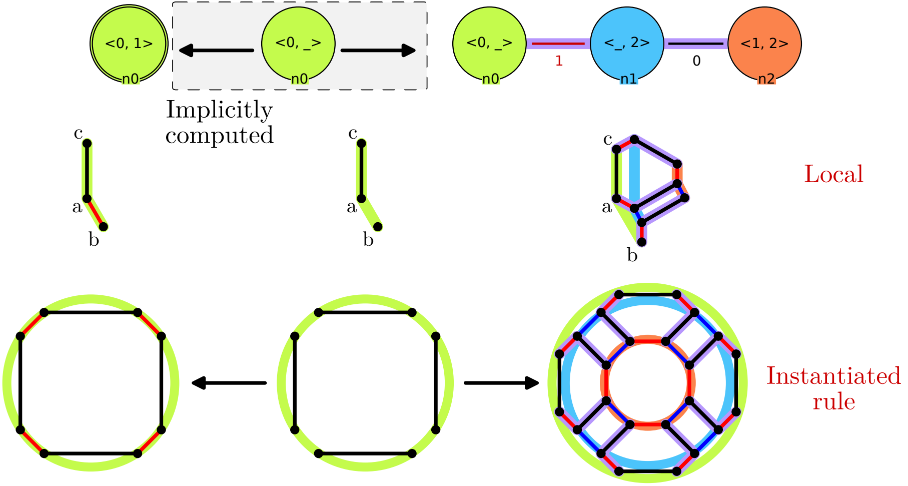
Orbit rewriting



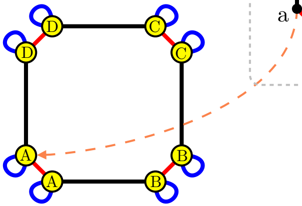


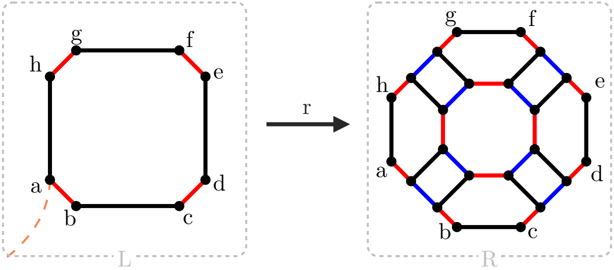
Orbit rewriting





Embedding expressions[1] (towards L)





Three families of expressions

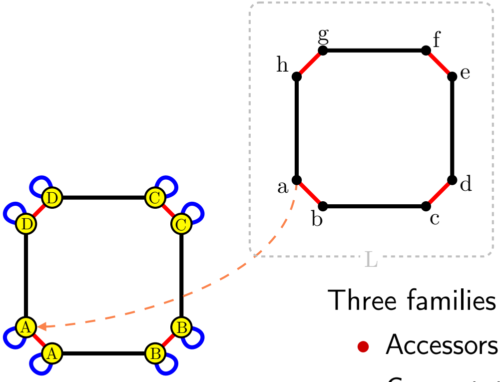
- Accessors

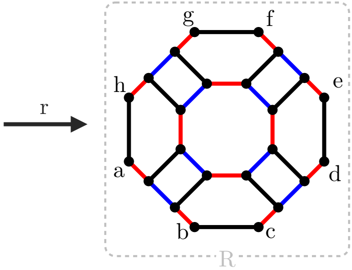
  a.color =

  a.position = A

[1]Bellet et al. 2017; Arnould et al. 2022.

Embedding expressions[1] (towards L)



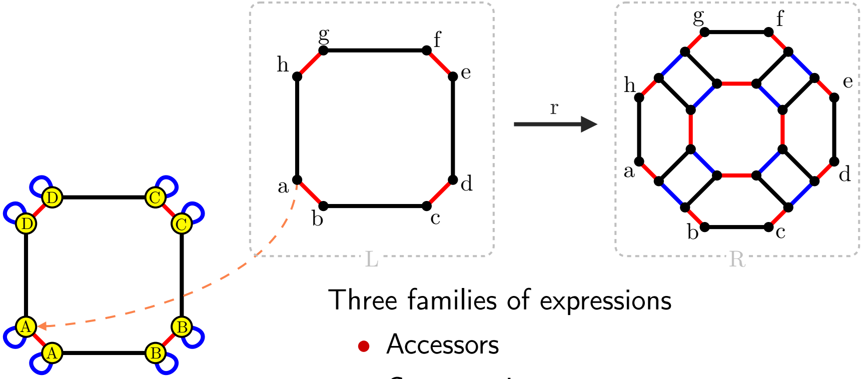


Three families of expressions

- Accessors
- Computations
  $\texttt{middle}(\{\color{red}{\bullet}, \color{green}{\bullet}\}) = \color{yellow}{\bullet}$

[1] Bellet et al. 2017; Arnould et al. 2022.

Embedding expressions[1] (towards L)

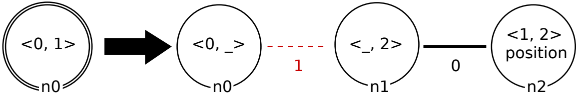


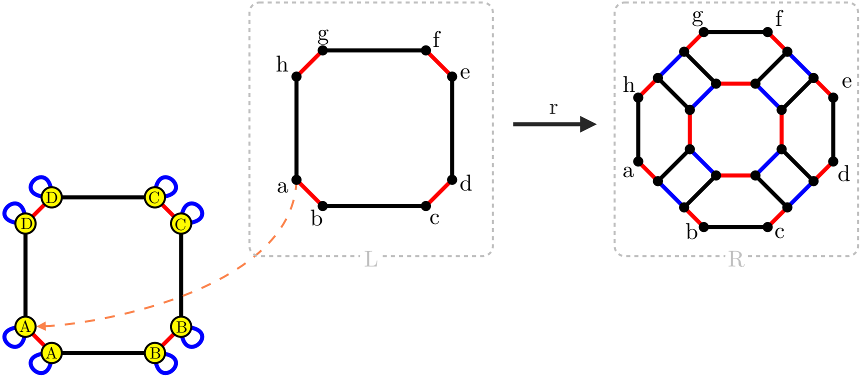
Three families of expressions

- Accessors
- Computations
- Gmap traversals
  a@0.position $= D$
  $\text{position}_{\langle 0,1\rangle}(a) = \{A, B, C, D\}$

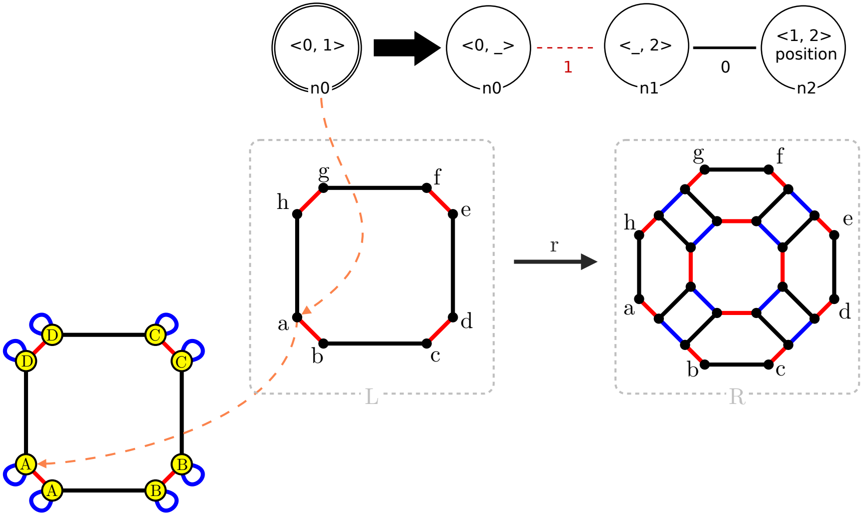
[1] Bellet et al. 2017; Arnould et al. 2022.

Extension to schemes

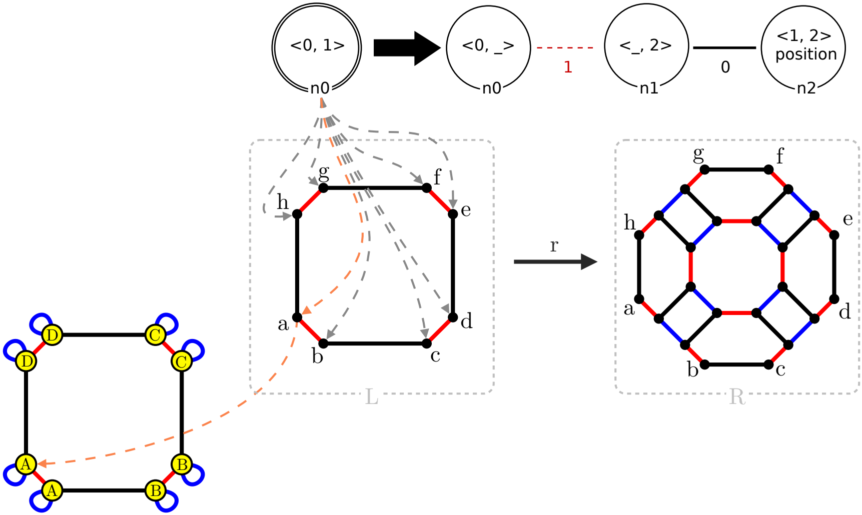




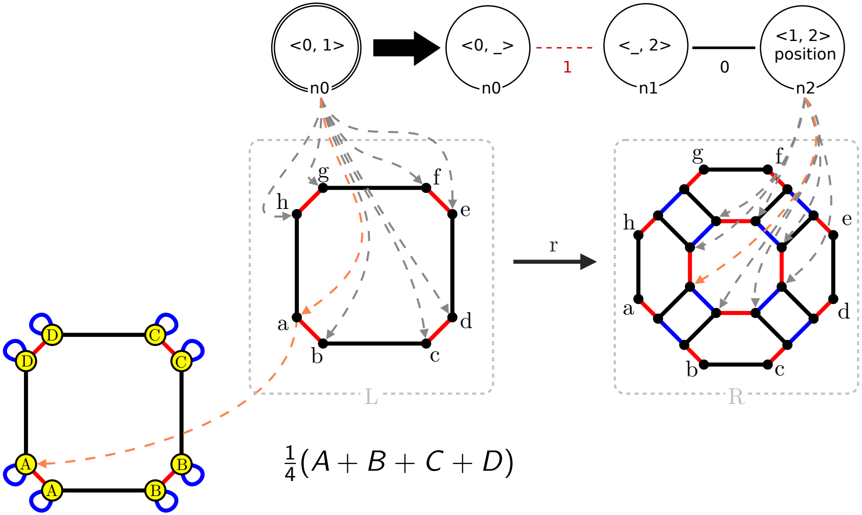
Extension to schemes



Extension to schemes

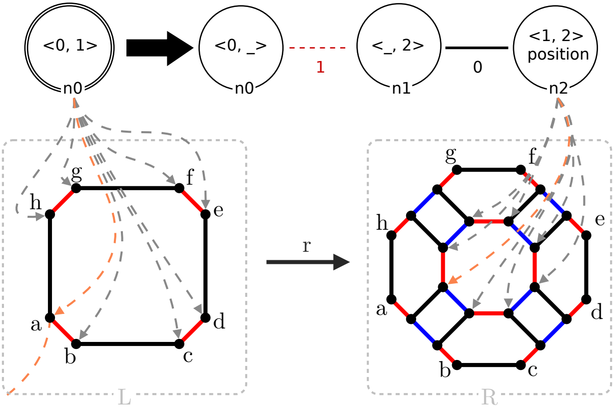


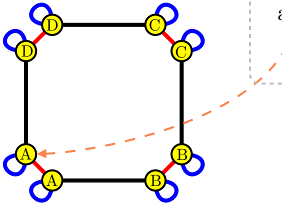
Extension to schemes



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

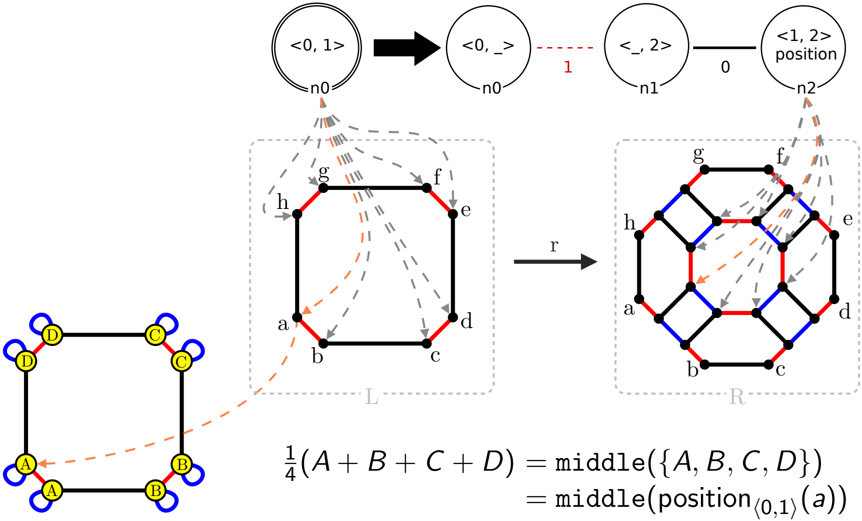
Extension to schemes





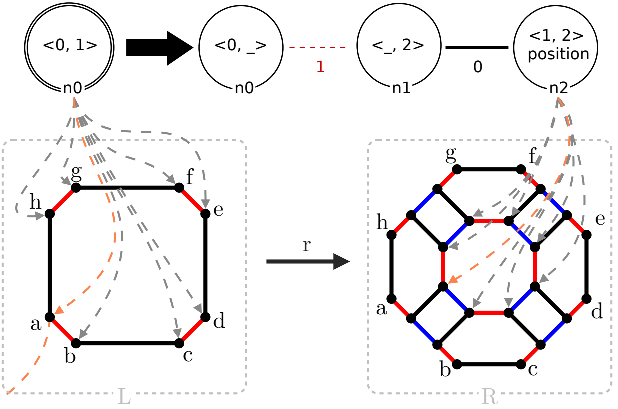
$$\frac{1}{4}(A + B + C + D) = \texttt{middle}(\{A, B, C, D\})$$

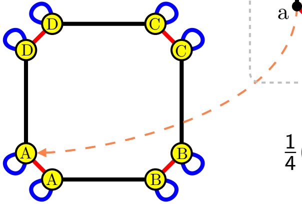
Extension to schemes



$$\frac{1}{4}(A+B+C+D) = \texttt{middle}(\{A, B, C, D\})$$
$$= \texttt{middle}(\text{position}_{\langle 0,1\rangle}(a))$$

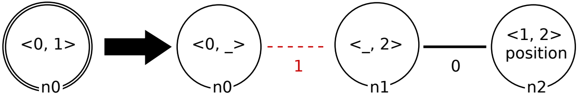
Extension to schemes





$$
\begin{aligned}
\tfrac{1}{4}(A + B + C + D) &= \mathtt{middle}(\{A, B, C, D\}) \\
&= \mathtt{middle}(\mathsf{position}_{\langle 0,1\rangle}(a)) \\
&= \mathtt{middle}(\mathsf{position}_{\langle 0,1\rangle}(n0))
\end{aligned}
$$

Extension to schemes

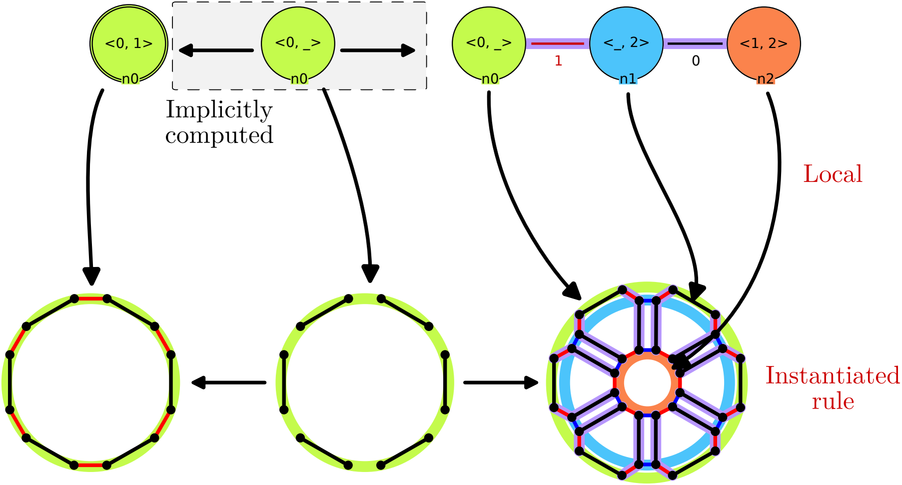


n2.position :
middle(position$_{<0,1>}$(n0))

Inferring geometric expressions

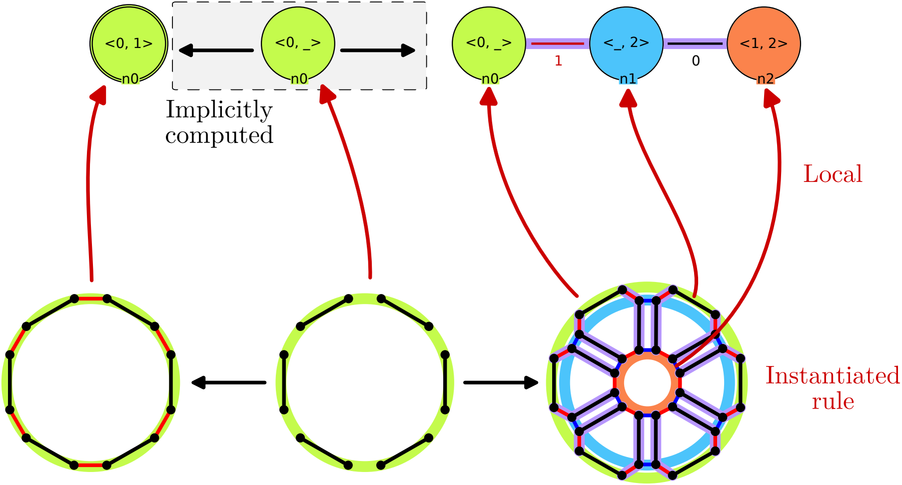
- How to retrieve the embedding computation expressions?

Topological folding algorithm[1]

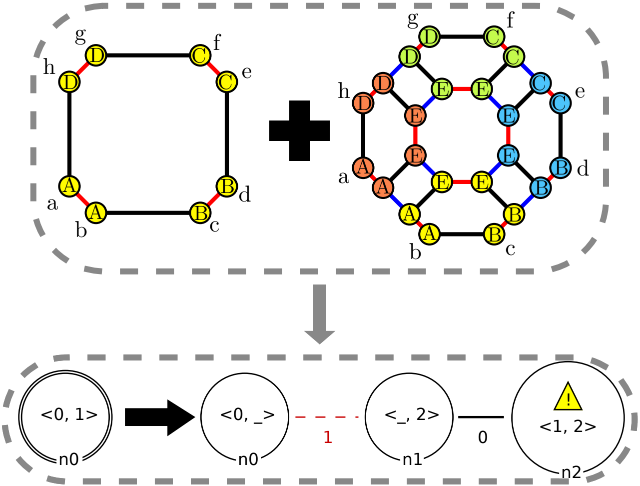


[1] Pascual et al. 2022.

Topological folding algorithm[1]

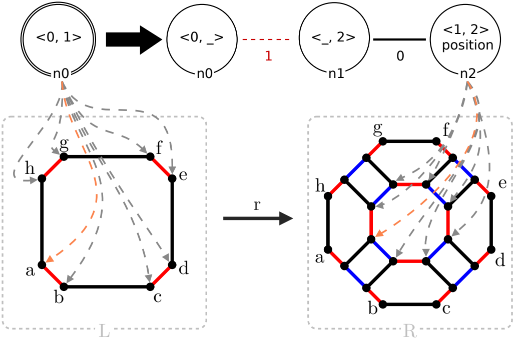


[1] Pascual et al. 2022.

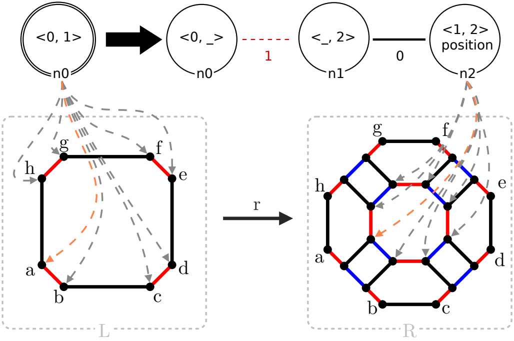


Embedding expressions are missing!

Need for abstraction on schemes

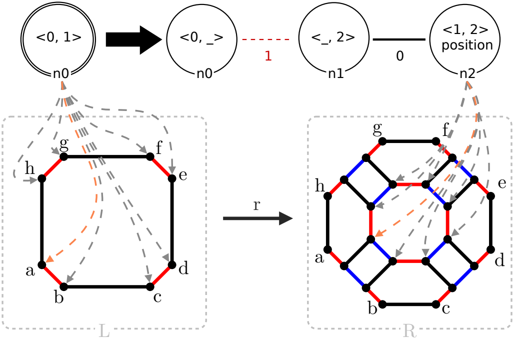


Need for abstraction on schemes



Schemes induce a topological abstraction

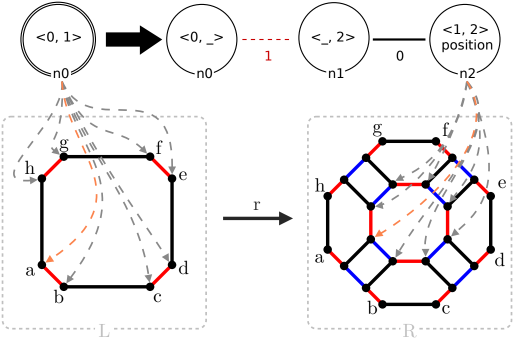
Need for abstraction on schemes



Schemes induce a topological abstraction

Issue: darts in the Gmap share the same expression

Need for abstraction on schemes

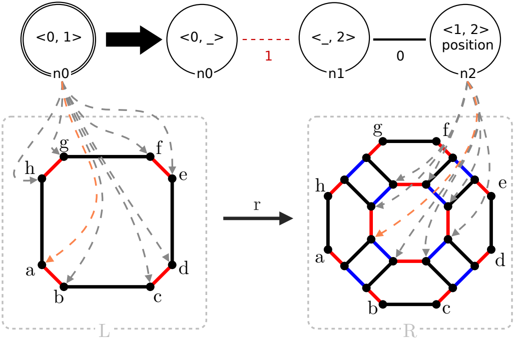


Schemes induce a topological abstraction

Issue: darts in the Gmap share the same expression

Solution: Exploit the topology

Need for abstraction on schemes



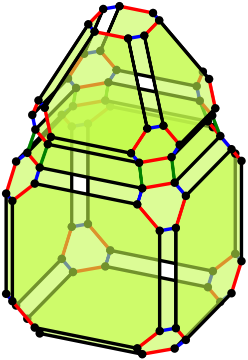
Schemes induce a topological abstraction

Issue: darts in the Gmap share the same expression

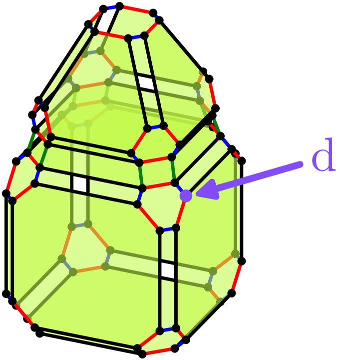
Solution: Exploit the topology

Points of interest

Points of interest



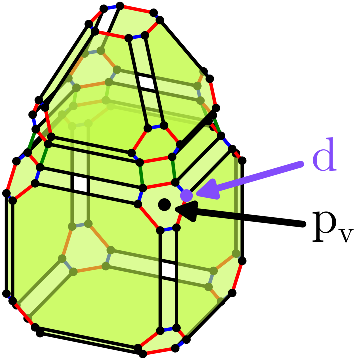
Points of interest



Points of interest

with

- p_v : vertex

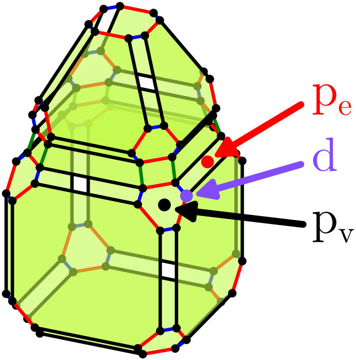


$$p_v = \texttt{middle}(\mathsf{position}_{\langle\rangle}(d))$$

Points of interest

with

- p_v : vertex
- p_e : edge midpoint

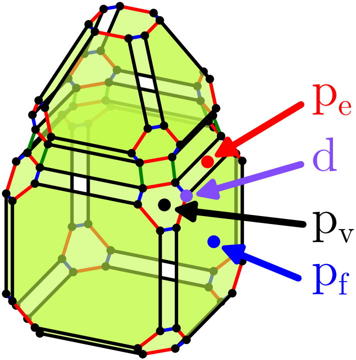


$$p_e = \texttt{middle}(\text{position}_{\langle 0 \rangle}(d))$$

Points of interest

with

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

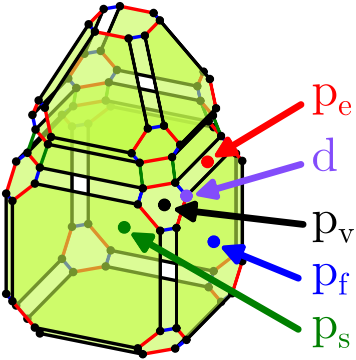


$$p_f = \texttt{middle}(\text{position}_{\langle 0,1\rangle}(d))$$

Points of interest

with

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

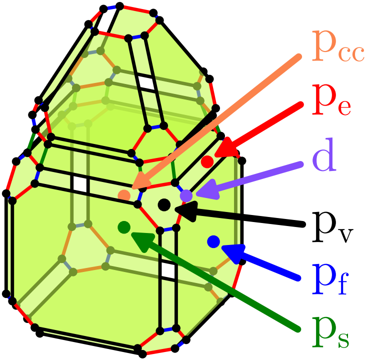


$$p_s = \texttt{middle}(\text{position}_{\langle 0,1,2\rangle}(d))$$

Points of interest

with

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

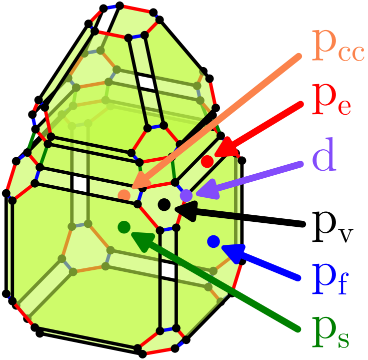


$$p_{cc} = \texttt{middle}(\text{position}_{\langle 0,1,2,3\rangle}(d))$$

Points of interest

with

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



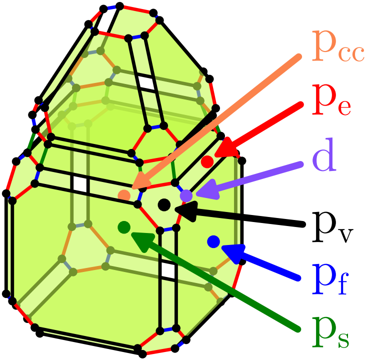
Looking for

$$f(p_v, p_e, p_f, p_s, p_{cc})$$

Points of interest

with

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

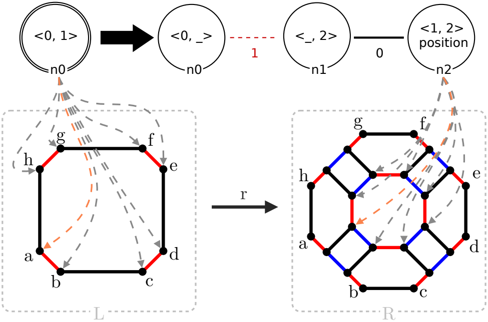


Looking for

$$f(p_v, p_e, p_f, p_s, p_{cc}) = w_v p_v + w_e p_e + w_f p_f + w_s p_s + w_{cc} p_{cc} + t$$

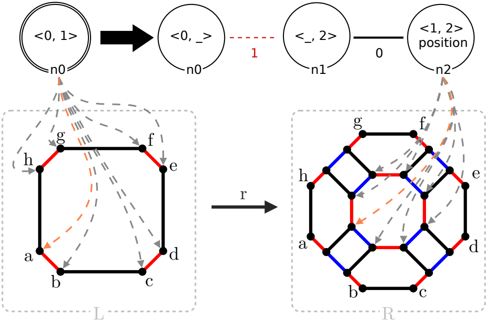
L is the set of affine expressions over the points of interest

Building the logical specification



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

Building the logical specification

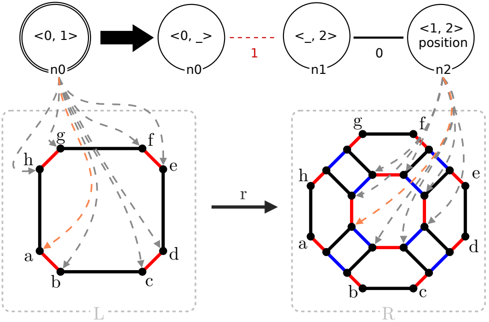


The position expression of $n2$ only depends on $n0$

Symbolic equation

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

Building the logical specification



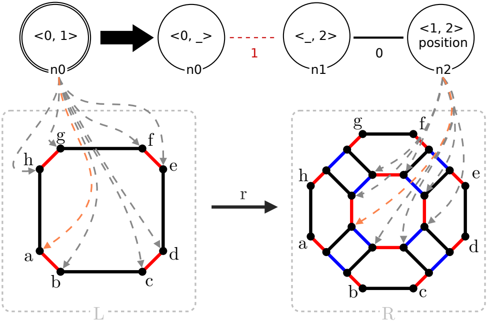
Symbolic equation

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

- One equation per dart (8 darts).

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

Building the logical specification



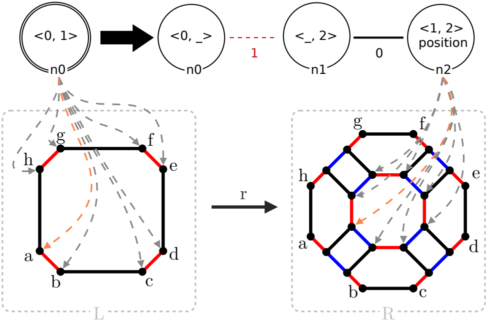
The position expression of $n2$ only depends on $n0$

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

Symbolic equation

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

Building the logical specification



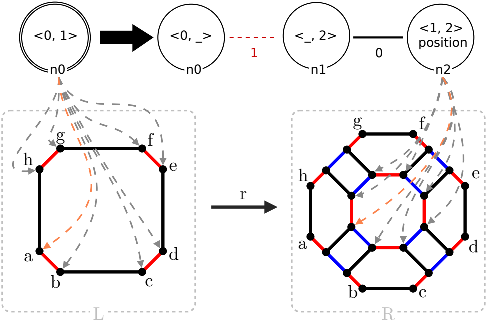
The position expression of $n2$ only depends on $n0$

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

Symbolic equation

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

Building the logical specification

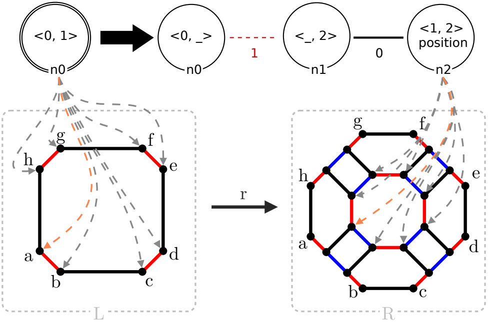


The position expression of $n2$ only depends on $n0$

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

φ is the concrete system induced by the input-output example

Building the logical specification



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.

φ is the concrete system induced by the input-output example

Solved via an SMT solver (Z3, OR-Tools)

Solving the barycentric triangulation

Symbolic 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

Symbolic 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

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

Solving the barycentric triangulation

Symbolic 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

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

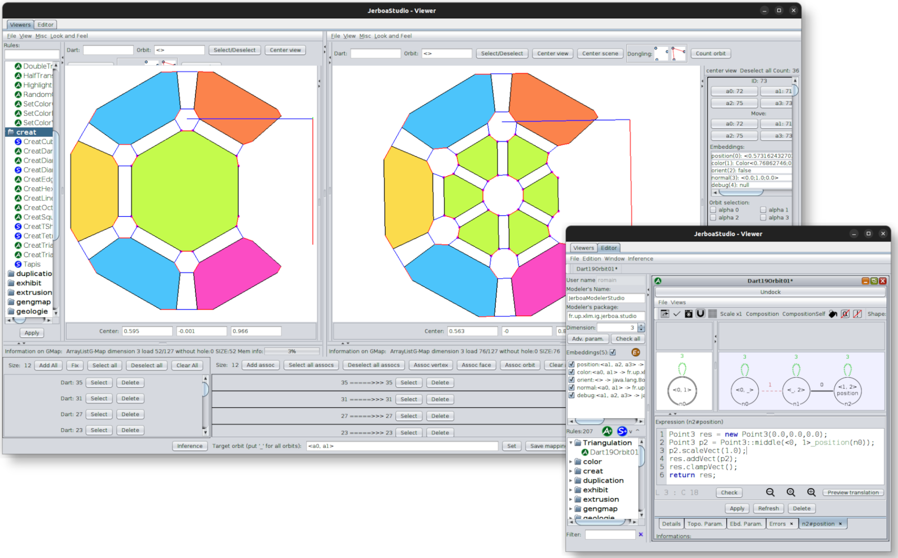
Solution found

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

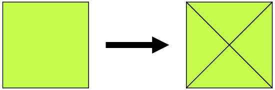
JerboaStudio and applications

- Implementation in Jerboa

JerboaStudio

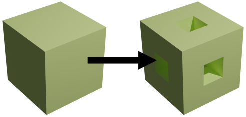


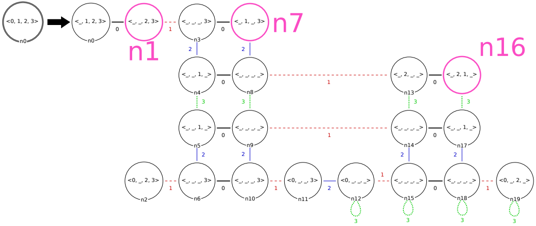
Generated code for the triangulation

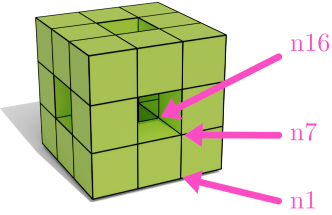


```
// no translation
Point3 res = new Point3(0.0,0.0,0.0);

// face
Point3 p2 = Point3::middle(<0,1>_position(n0));

// weight
p2.scale(1.0);

// added to the result
res.addVect(p2);

// return the value
return res;
```

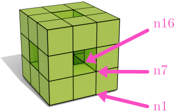
Menger Sponge







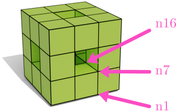
Menger Sponge



Node $n1$

```
Point3 res = new Point3(0.0,0.0,0.0);
Point3 p0 = Point3::middle(<>_position(n0));
p0.scale(0.333333134651184);
res.addVect(p0);
Point3 p1 = Point3::middle(<0>_position(n0));
p1.scale(0.6666666865348816);
res.addVect(p1);
return res;
```

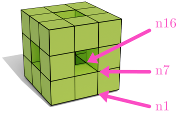
Menger Sponge



Node *n7*

```
Point3 res = new Point3(0.0,0.0,0.0);
Point3 p0 = Point3::middle(<>_position(n0));
p0.scale(0.333333134651184);
res.addVect(p0);
Point3 p2 = Point3::middle(<0,1>_position(n0));
p2.scale(0.666666865348816);
res.addVect(p2);
return res;
```

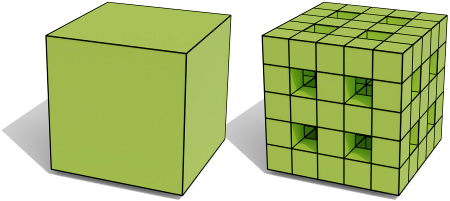
Menger Sponge



Node *n*16

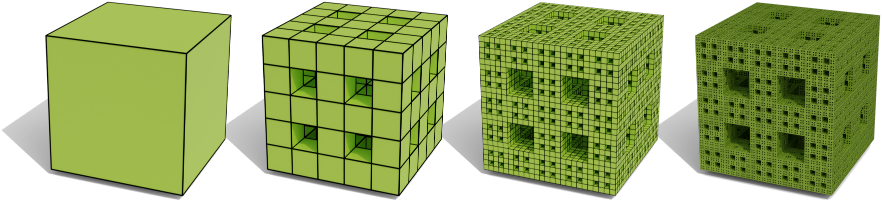
```
Point3 res = new Point3(0.0,0.0,0.0);
Point3 p0 = Point3::middle(<>_position(n0));
p0.scale(0.333333134651184);
res.addVect(p0);
Point3 p3 = Point3::middle(<0,1,2>_position(n0));
p3.scale(0.666666865348816);
res.addVect(p3);
return res;
```

(2, 2, 2)-Menger Polycube[1]



[1] Richaume et al. 2019.

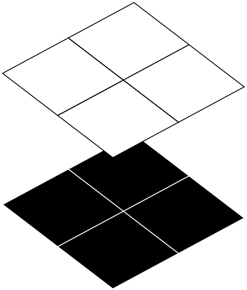
(2, 2, 2)-Menger Polycube[1]



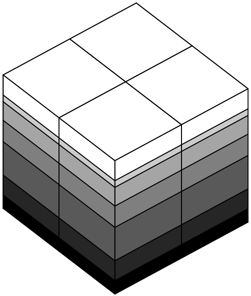
[1] Richaume et al. 2019.

Geology inspired

Before



After



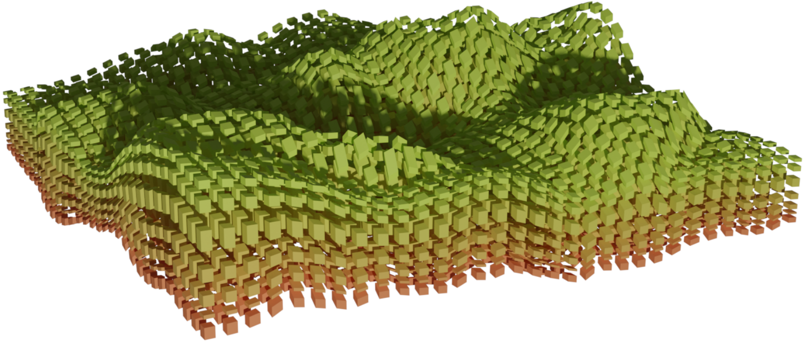
Positions and colors

Geology inspired

Before

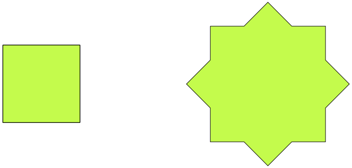
Geology inspired

After

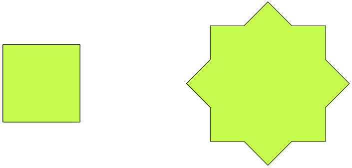


Limits

Von Koch's snowflake generated by L-systems

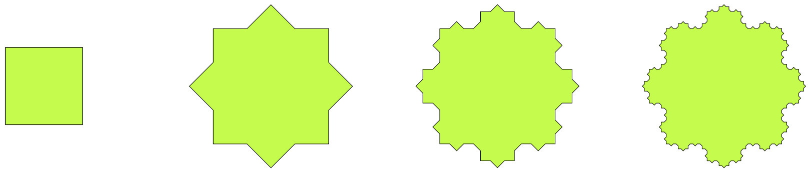


Inferred

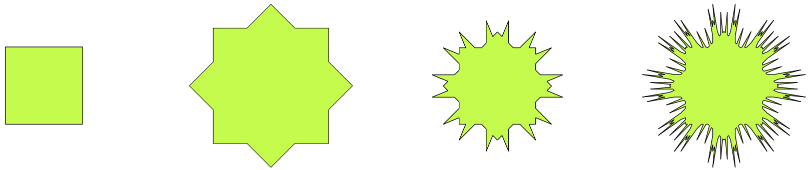


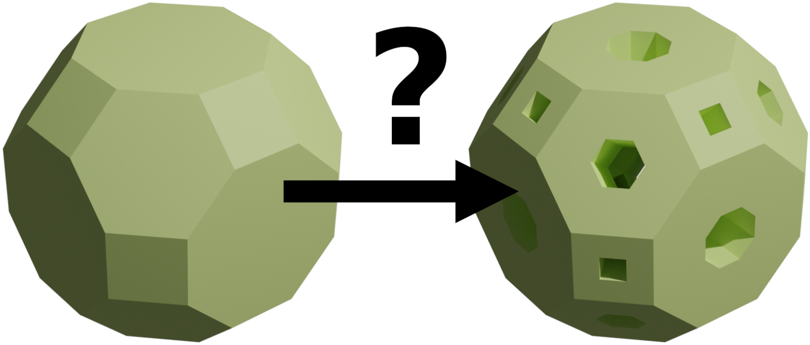
Limits

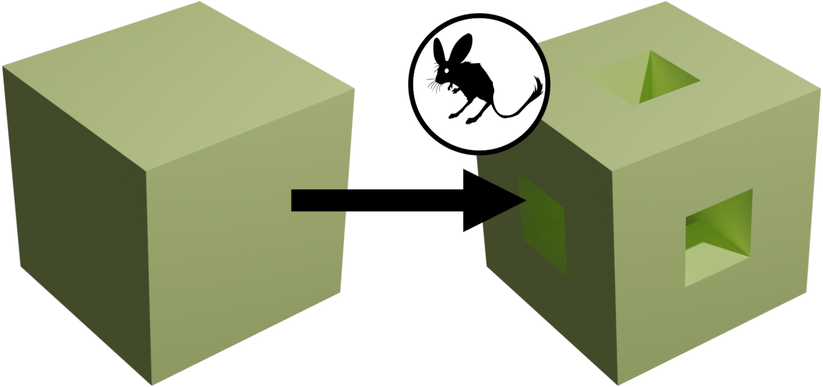
Von Koch's snowflake generated by L-systems



Inferred







Geometric inference and program synthesis

	L	φ
Rule level	Rule scheme	Instantiated rule

Geometric inference and program synthesis

	L	φ
Rule level	Rule scheme	Instantiated rule
Corresponds to	Affine combinations of points of interest	Concrete system derived from the example

Geometric inference and program synthesis

	L	φ
Rule level	Rule scheme	Instantiated rule
Corresponds to	Affine combinations of points of interest	Concrete system derived from the example
Property	Finite	Encodes redundancies

Geometric inference and program synthesis

	L	φ
Rule level	Rule scheme	Instantiated rule
Corresponds to	Affine combinations of points of interest	Concrete system derived from the example
Property	Finite	Encodes redundancies
Extend with	• other points of interest • other computations	• multi-examples • counter-examples

Geometric inference vs program synthesis?

Similarities

Geometric inference vs program synthesis?

Similarities

- Formal specification of the expected result

Geometric inference vs program synthesis?

Similarities

- Formal specification of the expected result
- DSL

Geometric inference vs program synthesis?

Similarities

- Formal specification of the expected result
- DSL
- Resolution delegated to a solver

Geometric inference vs program synthesis?

Similarities

- Formal specification of the expected result
- DSL
- Resolution delegated to a solver

Differences

Geometric inference vs program synthesis?

Similarities

- Formal specification of the expected result
- DSL
- Resolution delegated to a solver

Differences

- Pretreatment induced by the topology

Geometric inference vs program synthesis?

Similarities

- Formal specification of the expected result
- DSL
- Resolution delegated to a solver

Differences

- Pretreatment induced by the topology
- Exploit symmetries

Geometric inference vs program synthesis?

Similarities

- Formal specification of the expected result
- DSL
- Resolution delegated to a solver

Differences

- Pretreatment induced by the topology
- Exploit symmetries
- Not so easy to play with examples

Geometric inference vs program synthesis?

Similarities

- Formal specification of the expected result
- DSL
- Resolution delegated to a solver

Differences

- Pretreatment induced by the topology
- Exploit symmetries
- Not so easy to play with examples

Is there any added value in re-thinking in terms of program synthesis?

References I

- Alur, Rajeev et al. (Oct. 2013). "Syntax-guided synthesis". In: 2013 Formal Methods in Computer-Aided Design. 2013 Formal Methods in Computer-Aided Design, pp. 1–8. DOI: 10.1109/FMCAD.2013.6679385.
- Alur, Rajeev et al. (Nov. 20, 2018). "Search-based program synthesis". In: Communications of the ACM 61.12, pp. 84–93. ISSN: 0001-0782. DOI: 10.1145/3208071.
- Arnould, Agnès et al. (2022). "Preserving consistency in geometric modeling with graph transformations". In: Mathematical Structures in Computer Science. DOI: 10.1017/S0960129522000226.

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.
- 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.

References III

- 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.
- Gulwani, Sumit et al. (Aug. 1, 2012). "Spreadsheet data manipulation using examples". In: Communications of the ACM 55.8, pp. 97–105. ISSN: 0001-0782. DOI: 10.1145/2240236.2240260.
- 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 IV

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 V

Merrell, Paul (July 26, 2023). "Example-Based Procedural Modeling Using Graph Grammars". In: ACM Transactions on Graphics 42.4, 60:1–60:16. ISSN: 0730-0301. DOI: 10.1145/3592119. (Visited on 10/05/2023).

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

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.

References VI

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.

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.

References VII

- 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.
- 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.