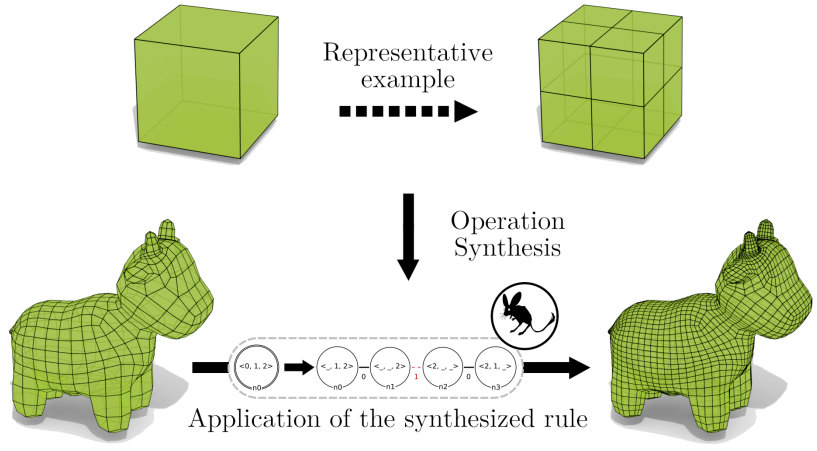
Program Synthesis for Geometric Modeling

Romain Pascual, Pascale Le Gall,
Hakim Belhaouari, and
Agnès Arnould

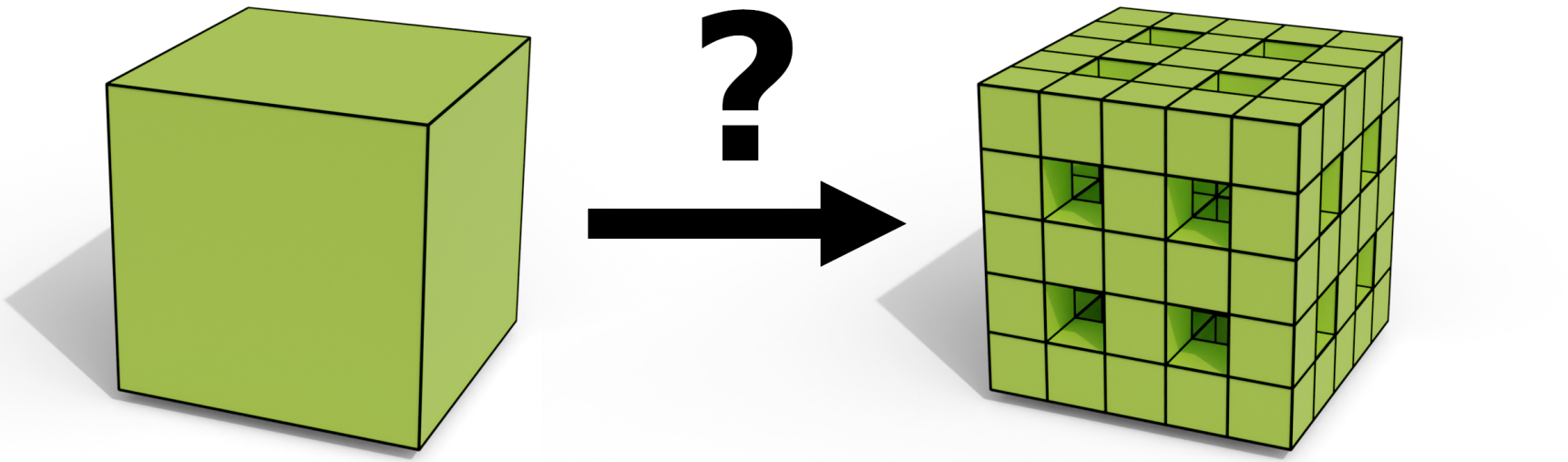


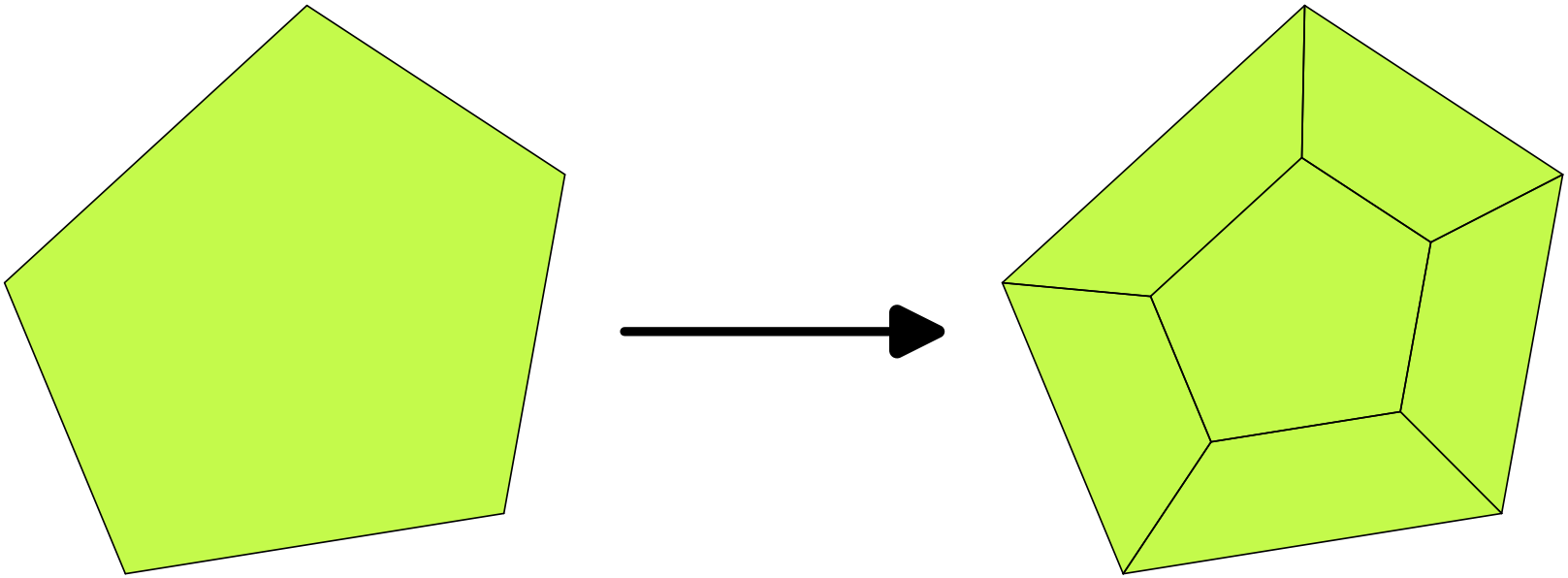
September 09, 2025

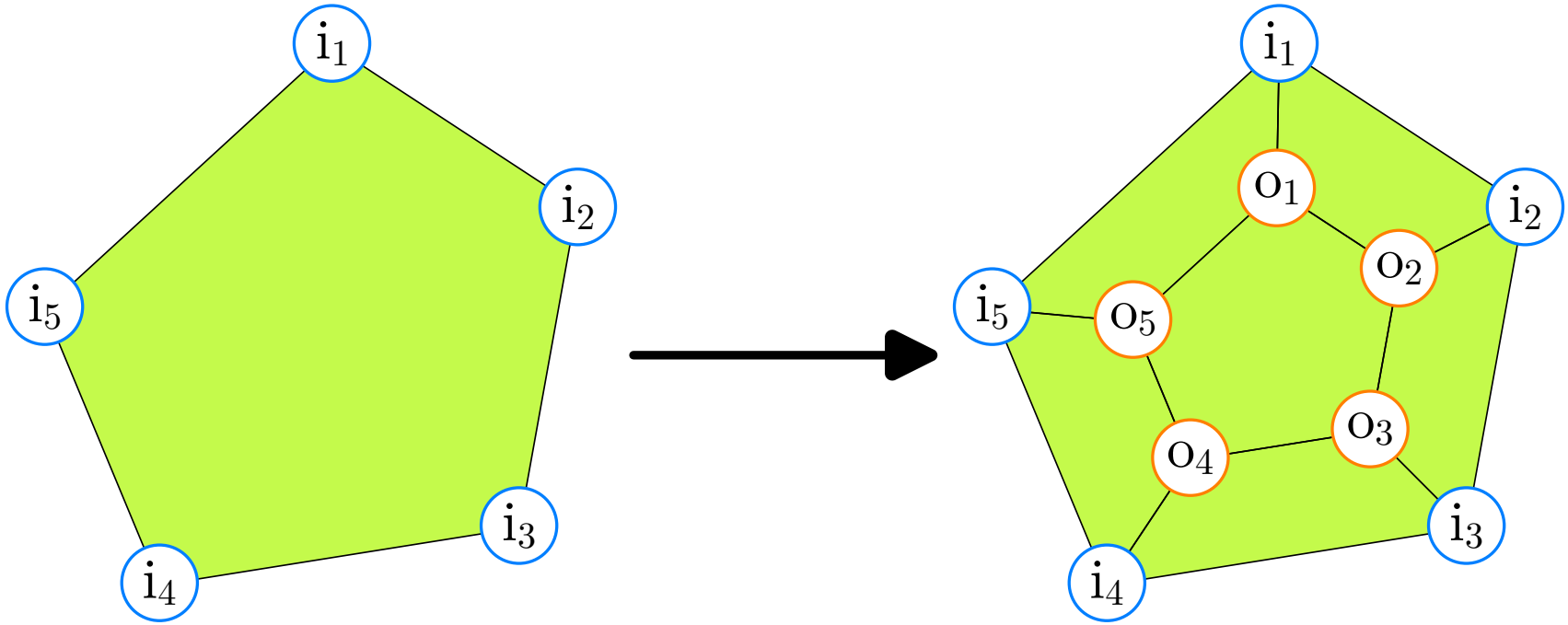
LOPSTR 2025

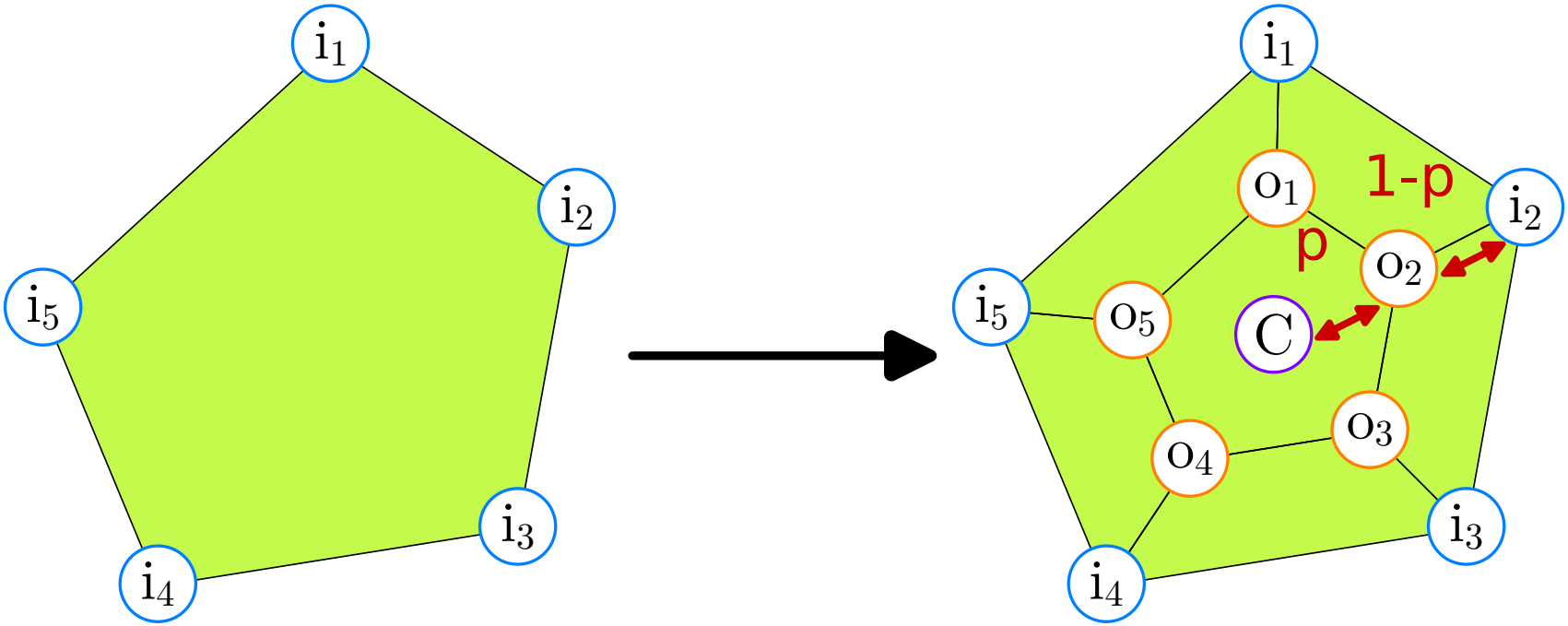
mics université PARIS-SACLAY xlim Université de Poitiers

Jerboa

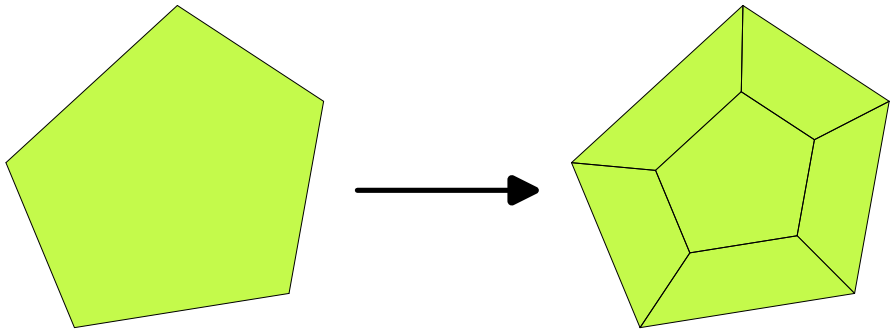




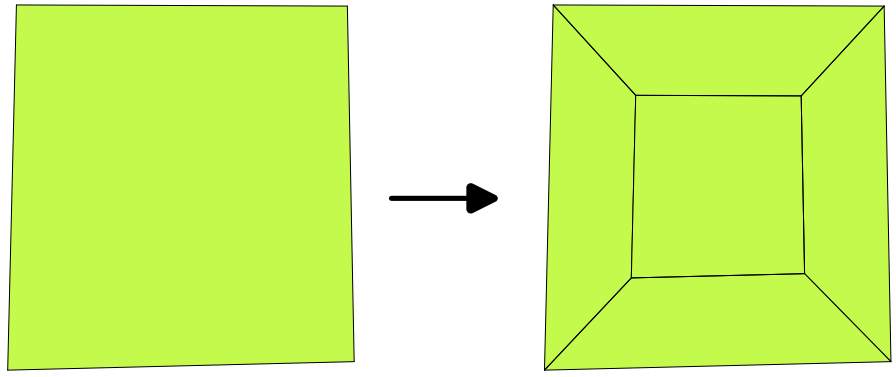

i1
i2
i3
i4
i5
i1
o1
i2
o2
i5
o5
o3
o4
i3
i4



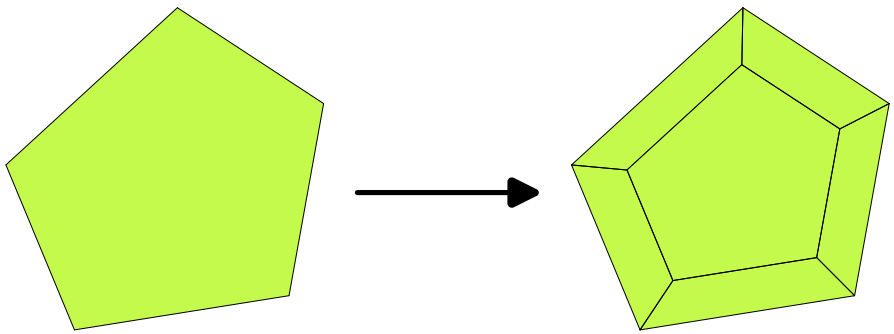
$$o_k = p \cdot i_k + (1 - p) \cdot \frac{1}{5} \sum_{n=1}^{5} i_n$$



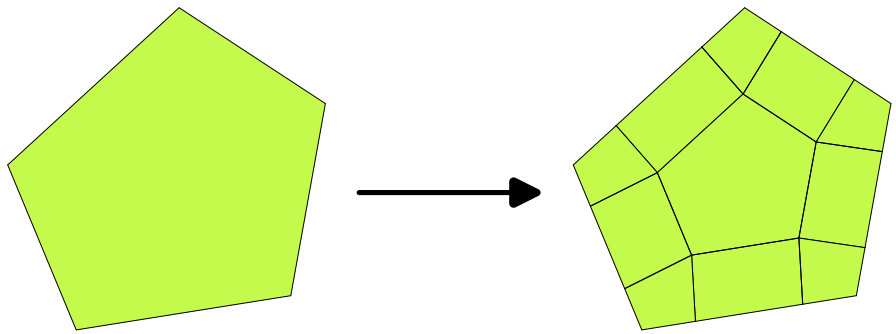
Input-to-output example



Different input topology

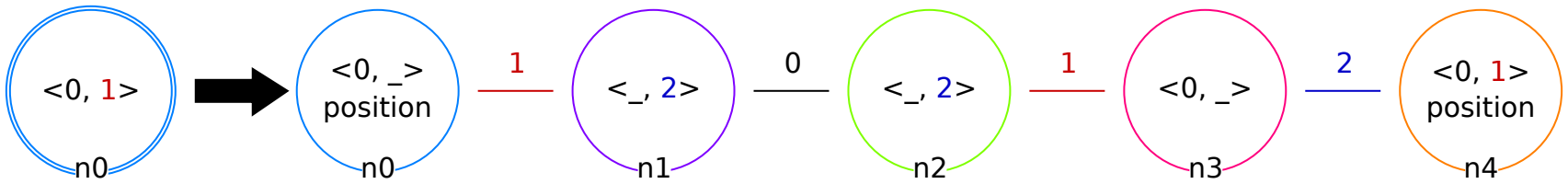


Different output geometry



Different operation

Jerboa



Position for n_0

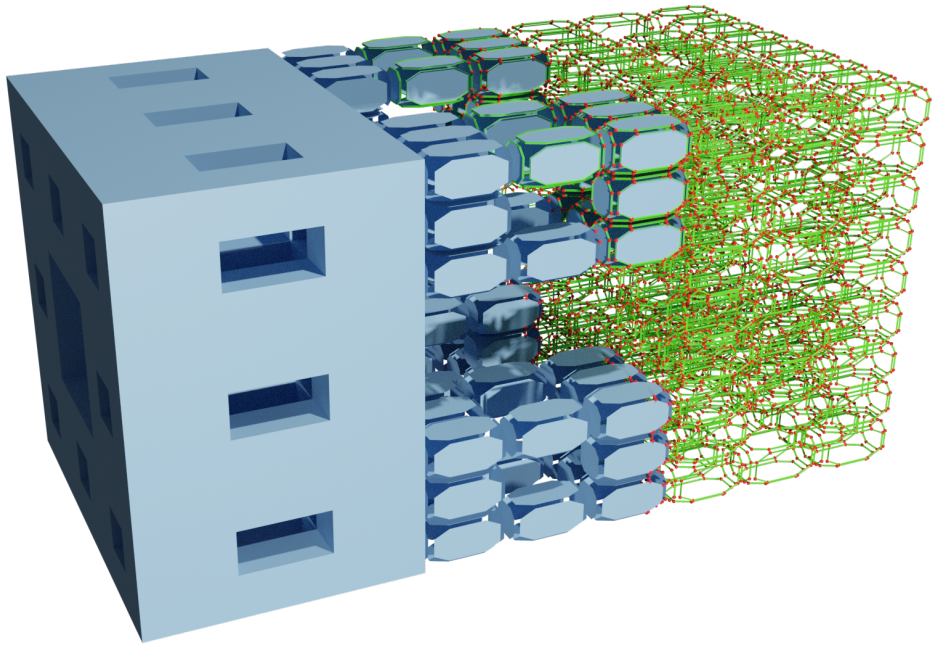
```
return n0.position;
```

Position for n_4

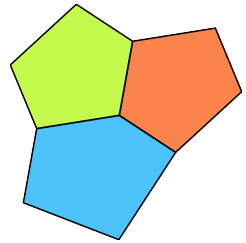
```
return Point3::midpt(
    Point3::middle(<0,1>_position(n0)),
    n0.position);
```

Embedded Generalized Maps

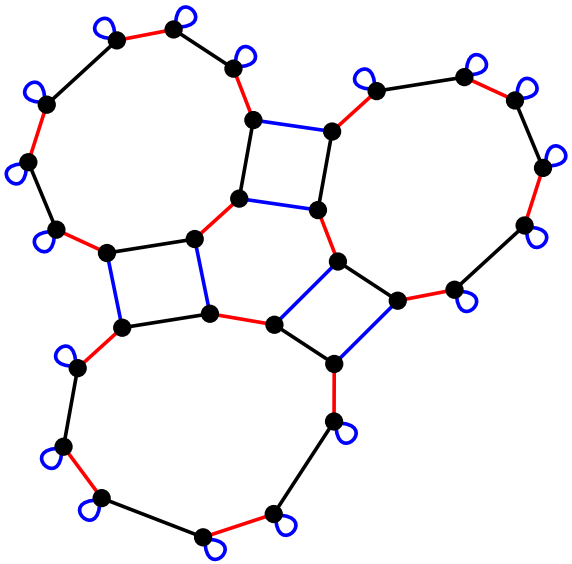
- How to represent objects?



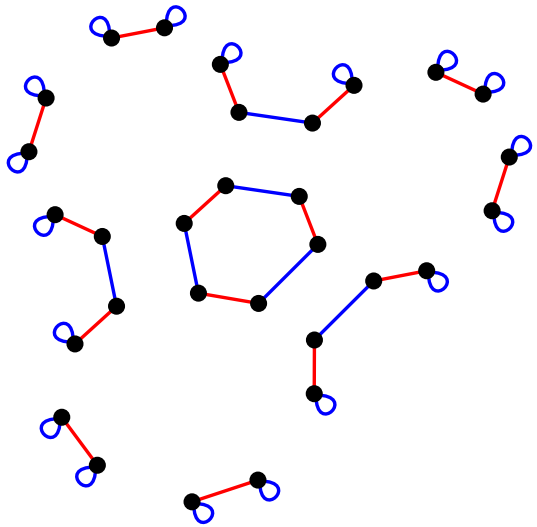
Generalized Maps[1] (Topology)



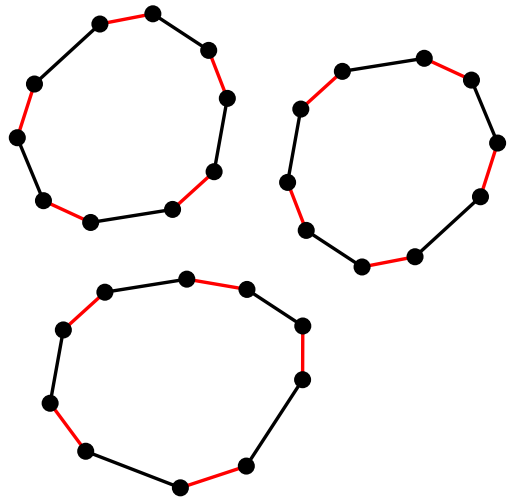
Orbit: Sub-graph induced by a subset $\langle o \rangle$ of dimensions



Legend: 0, 1, 2



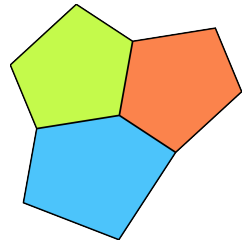
Vertices: orbits $\langle 1, 2 \rangle$



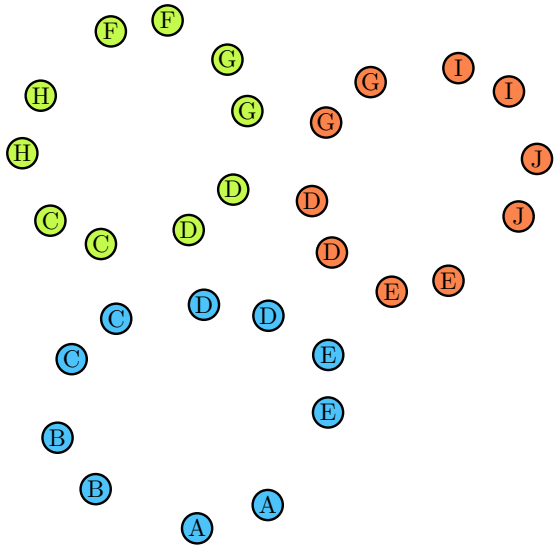
Faces: orbits $\langle 0, 1 \rangle$

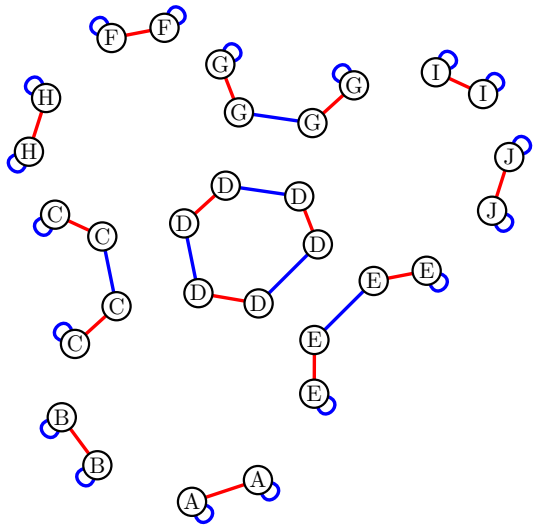
[1] Damiand et al. 2014.

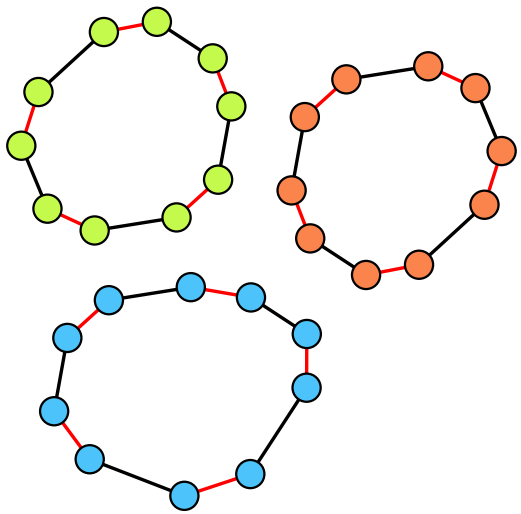
Embeddings (Geometry)



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



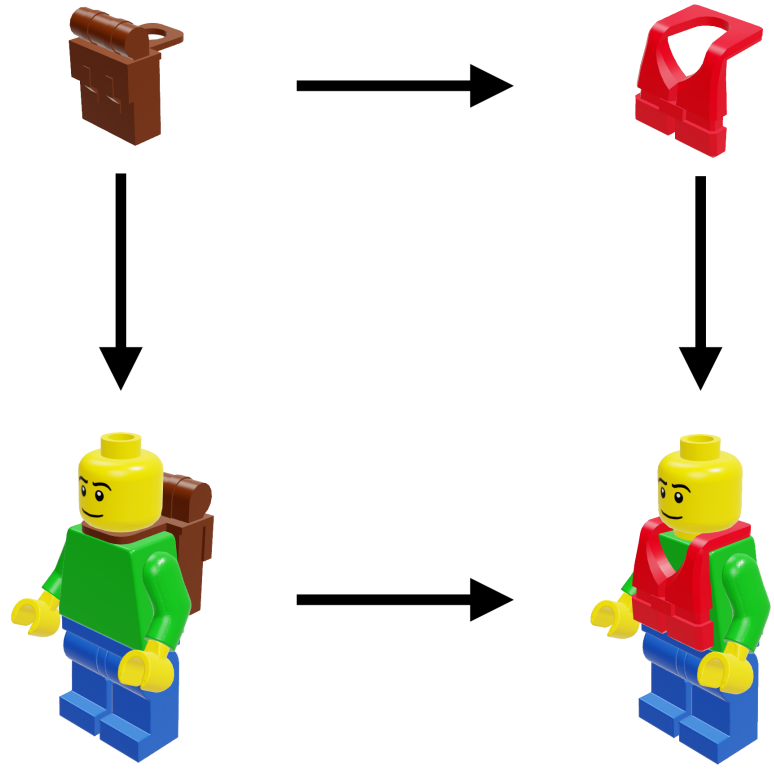




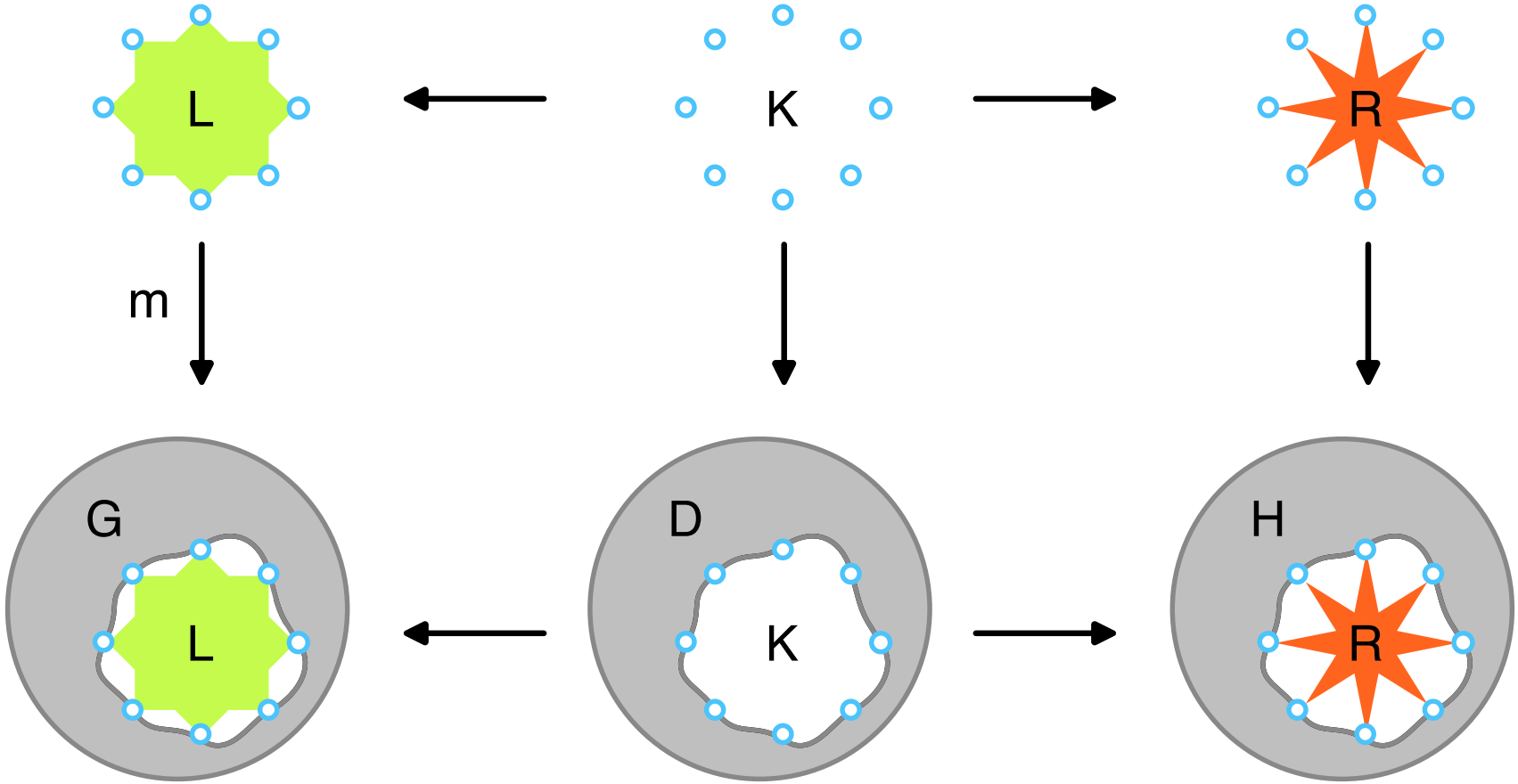
Legend: 0, 1, 2 `position` : $\langle 1, 2 \rangle \rightarrow$ `Point3` `color` : $\langle 0, 1 \rangle \rightarrow$ `ColorRGB`

Gmap Rewriting

- How to modify objects?

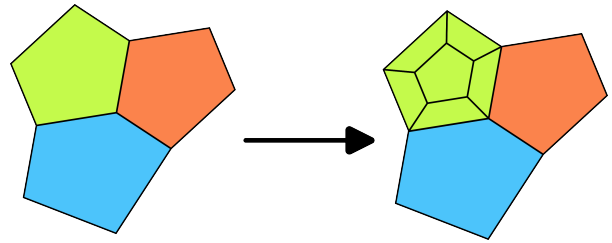


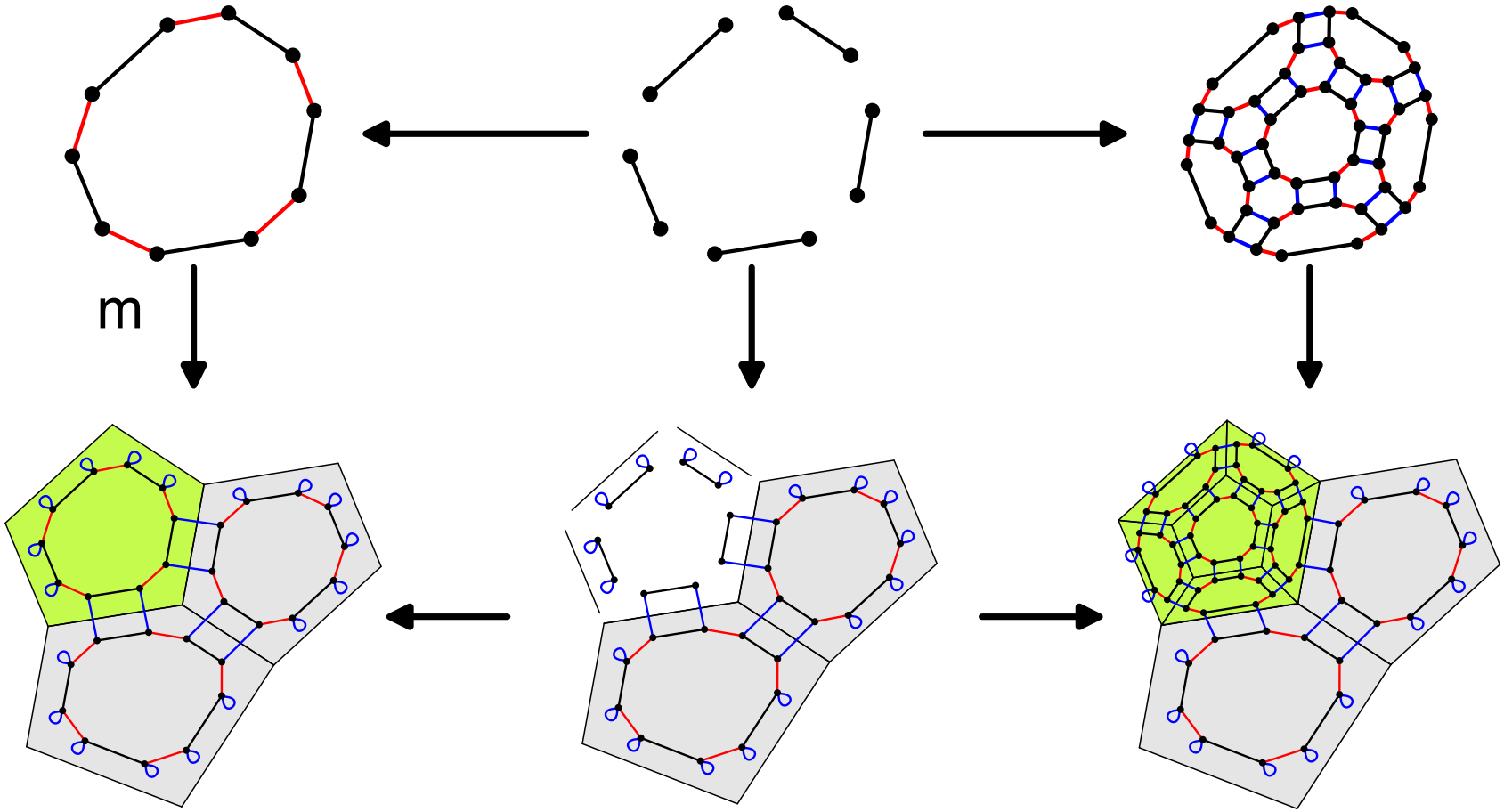
Graph Rewriting[1]



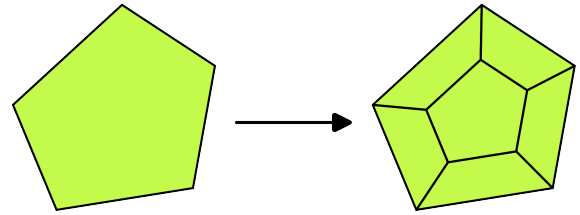
[1] Rozenberg 1997; Ehrig et al. 2006; Heckel et al. 2020.

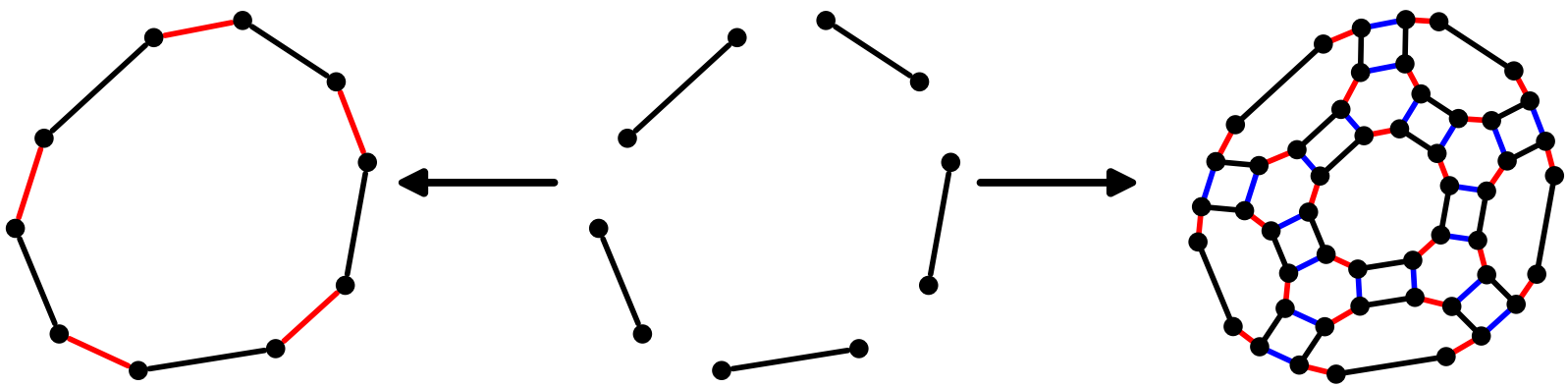
Gmap Rewriting





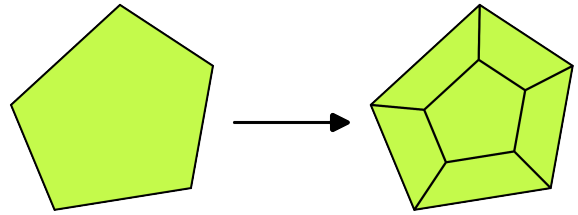
Orbit Rewriting[1]

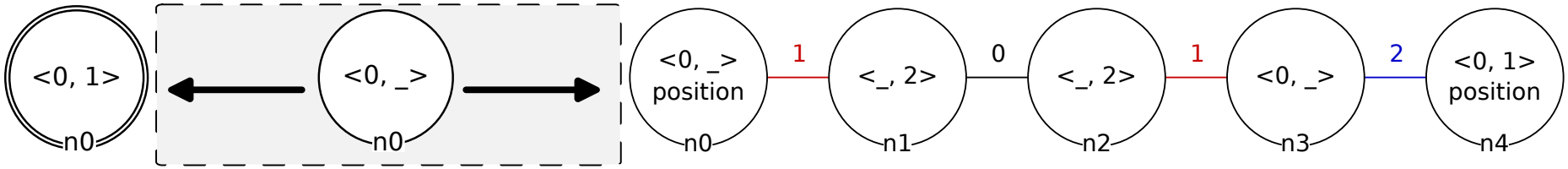




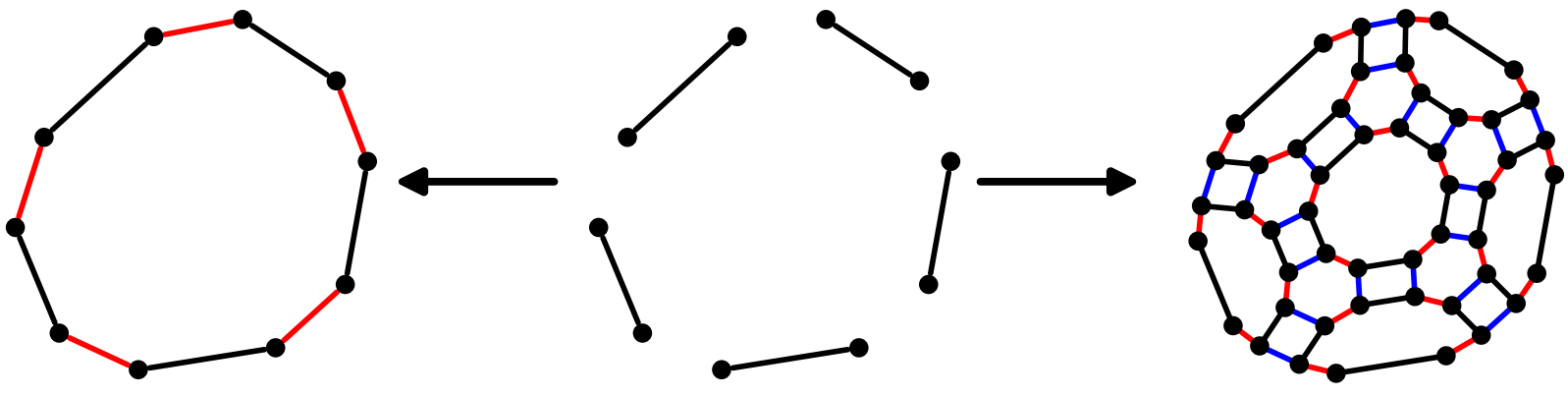
[1] Pascual et al. 2022b.

Orbit Rewriting[1]





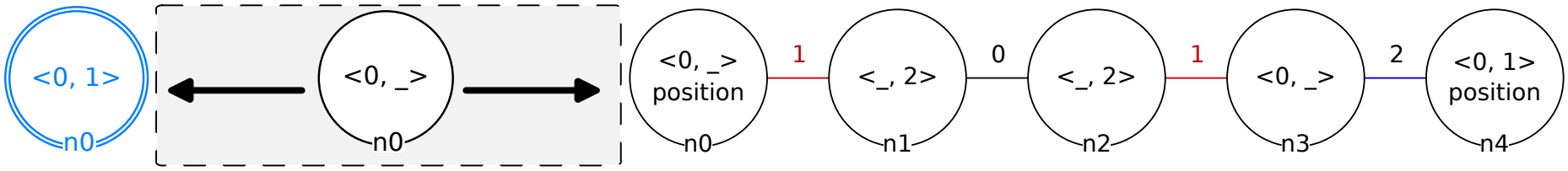
Implicilty Computed



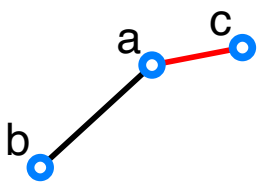
Instantiated Rule

[1] Pascual et al. 2022b.

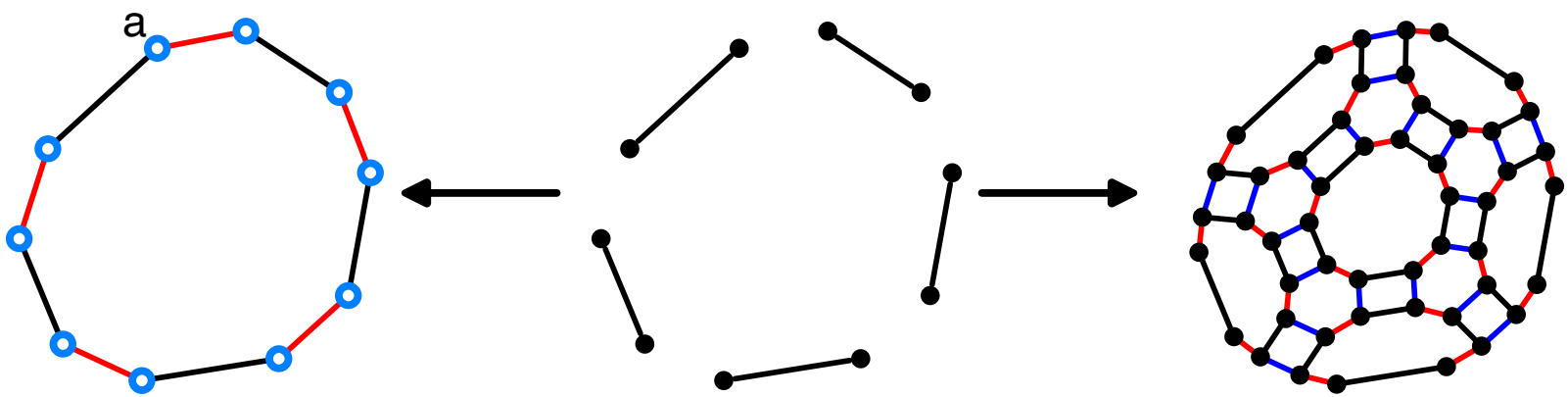
Orbit Rewriting[1]



Implicilty Computed



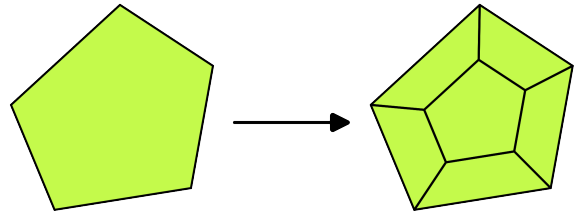
Local

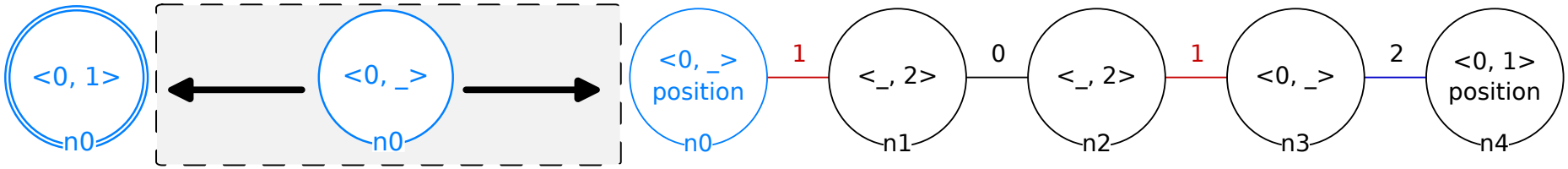


Instantiated Rule

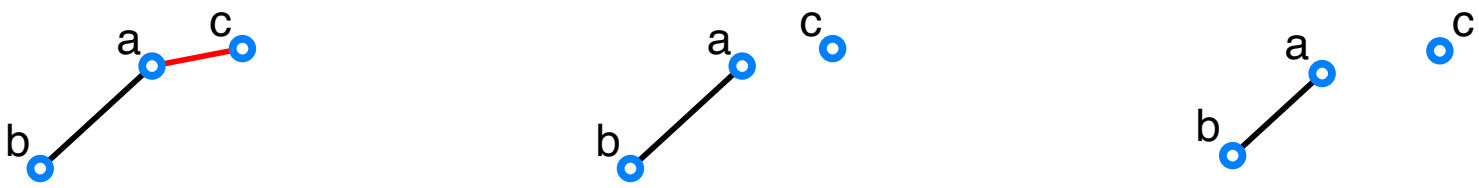
[1] Pascual et al. 2022b.

Orbit Rewriting[1]

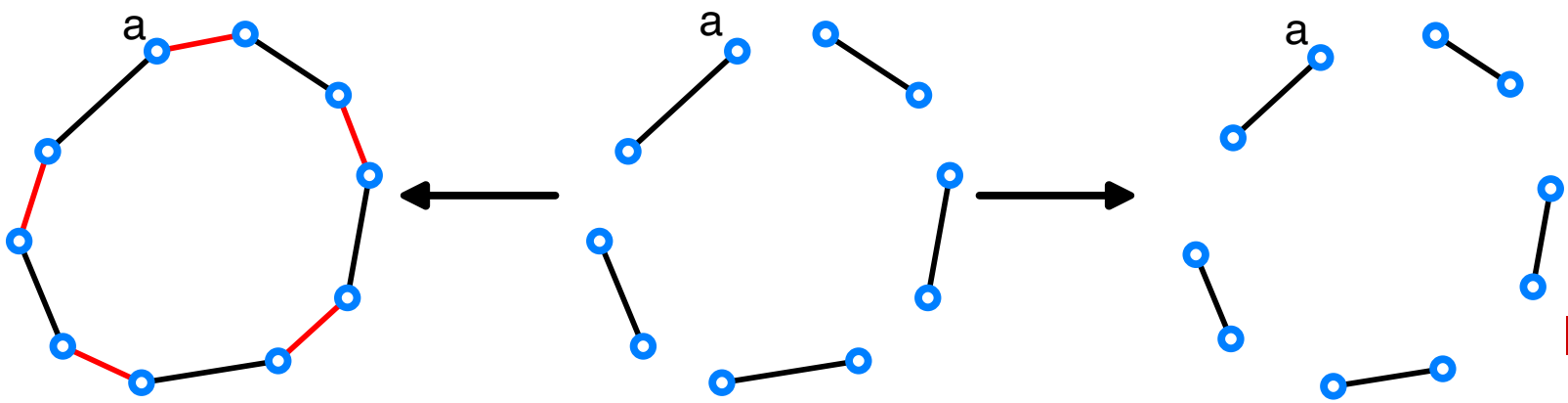




Implicilty Computed



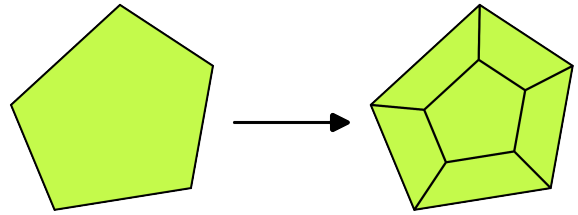
Local

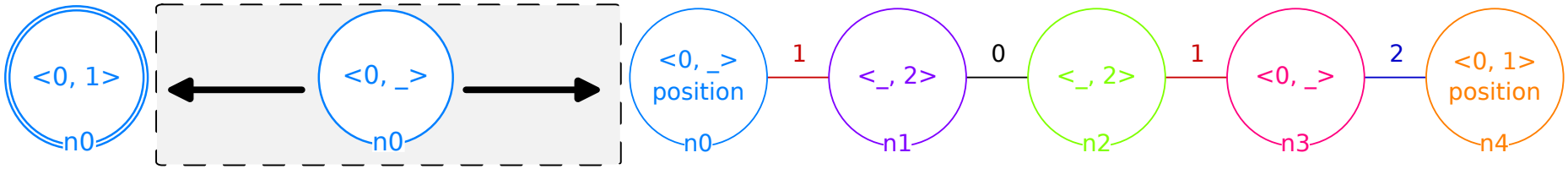


Instantiated Rule

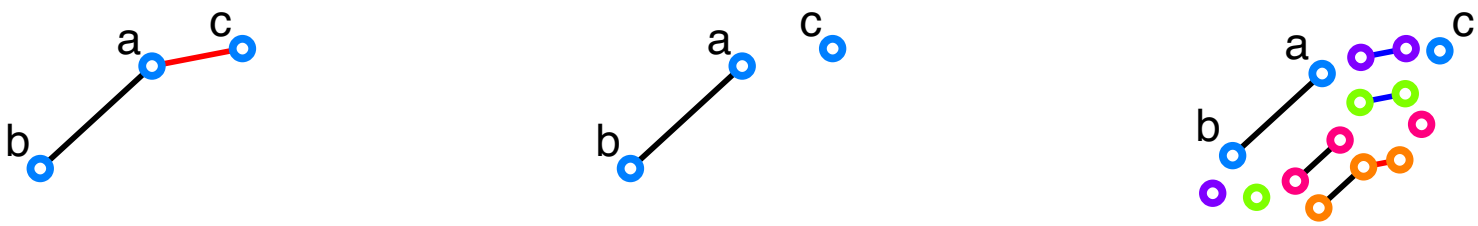
[1] Pascual et al. 2022b.

Orbit Rewriting[1]

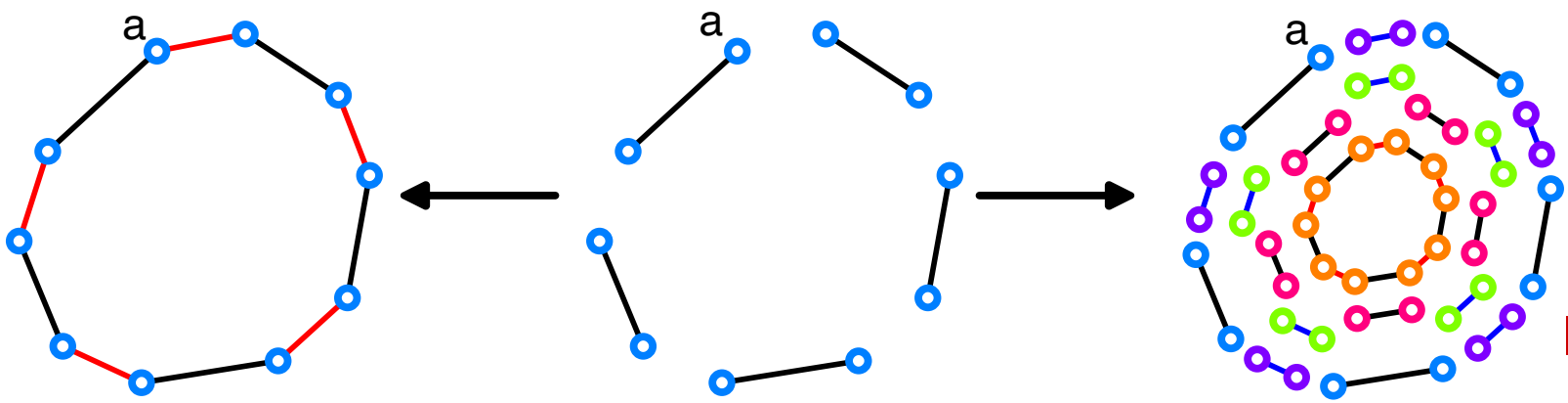




Implicilty Computed



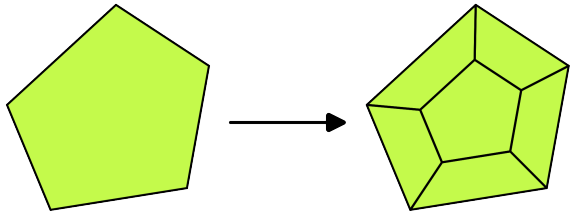
Local

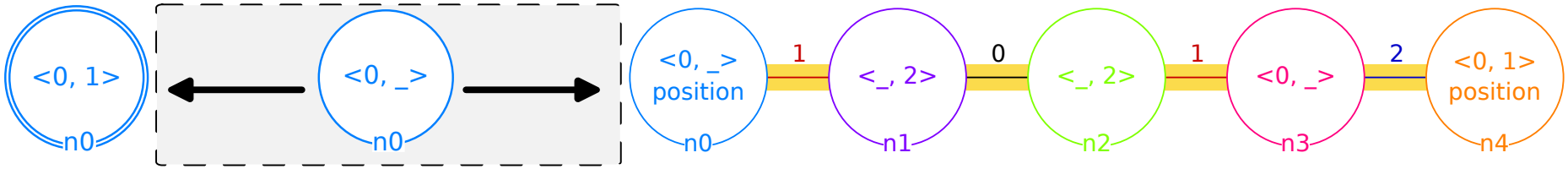


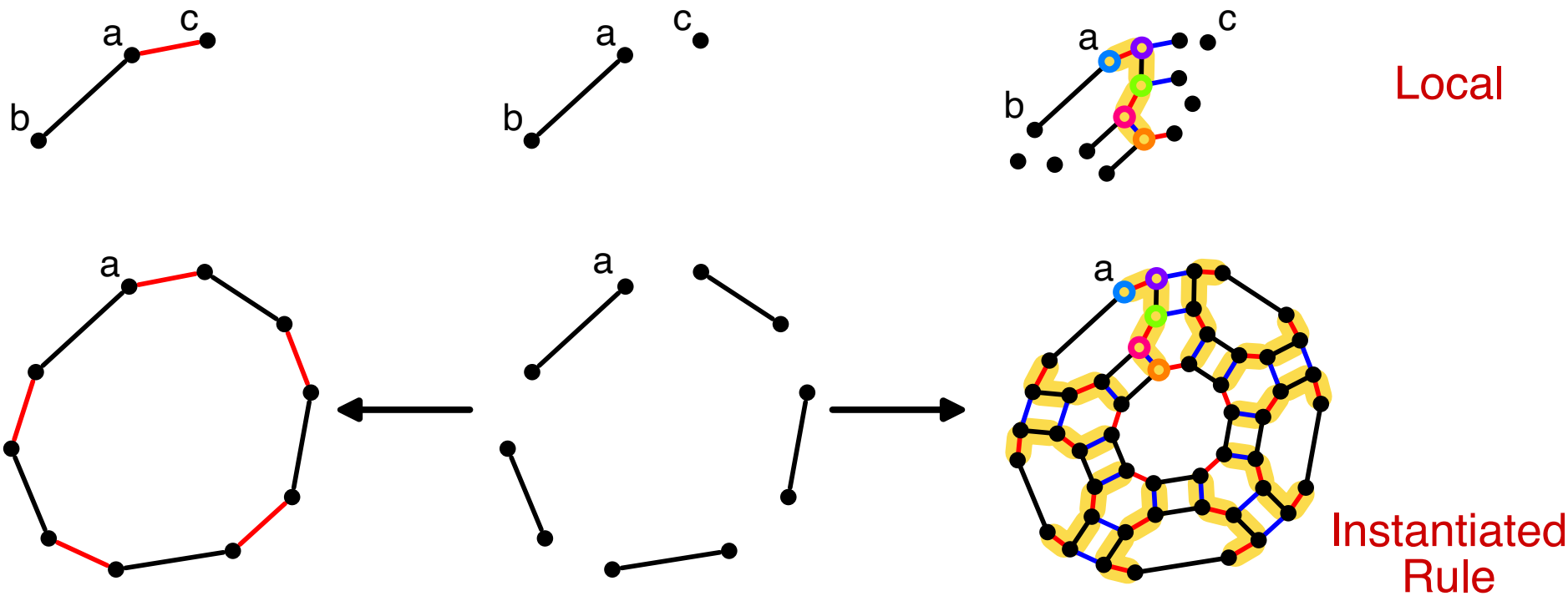
Instantiated Rule

[1] Pascual et al. 2022b.

Orbit Rewriting[1]

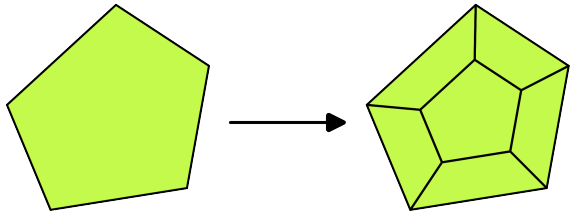


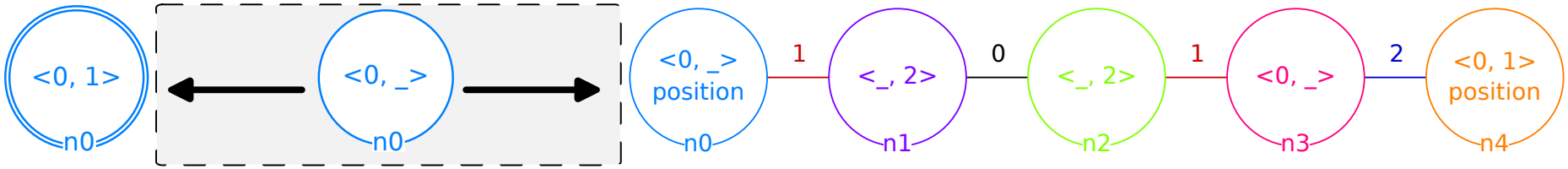




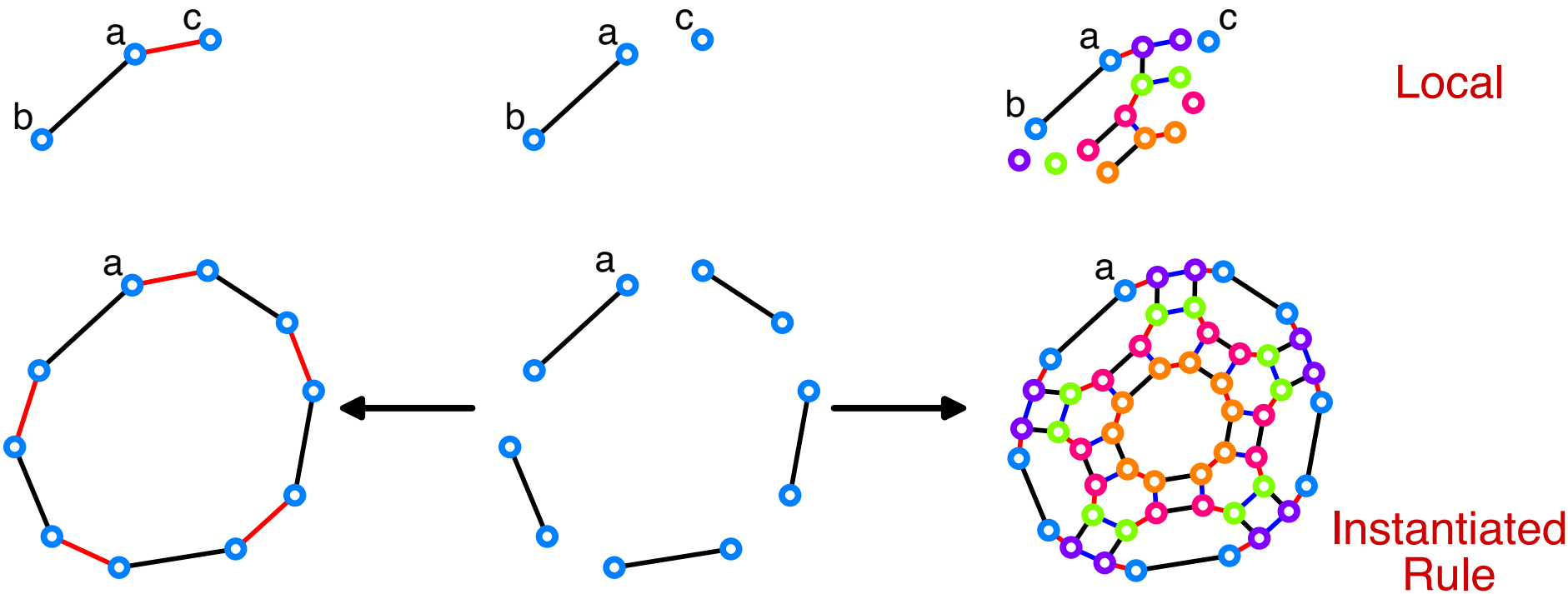
[1] Pascual et al. 2022b.

Orbit Rewriting[1]



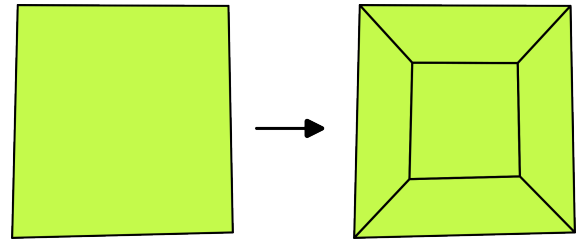


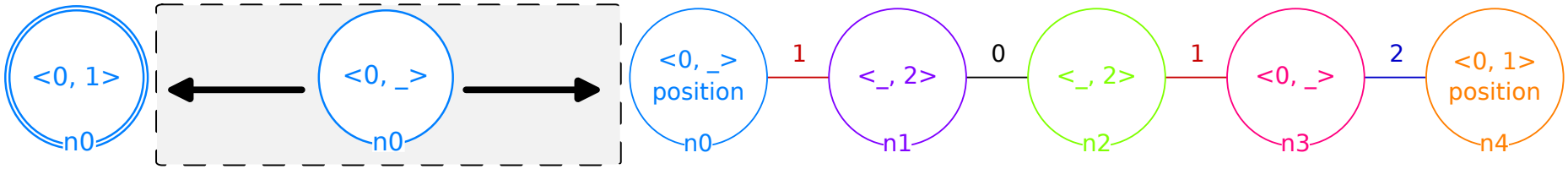
Implicilty Computed



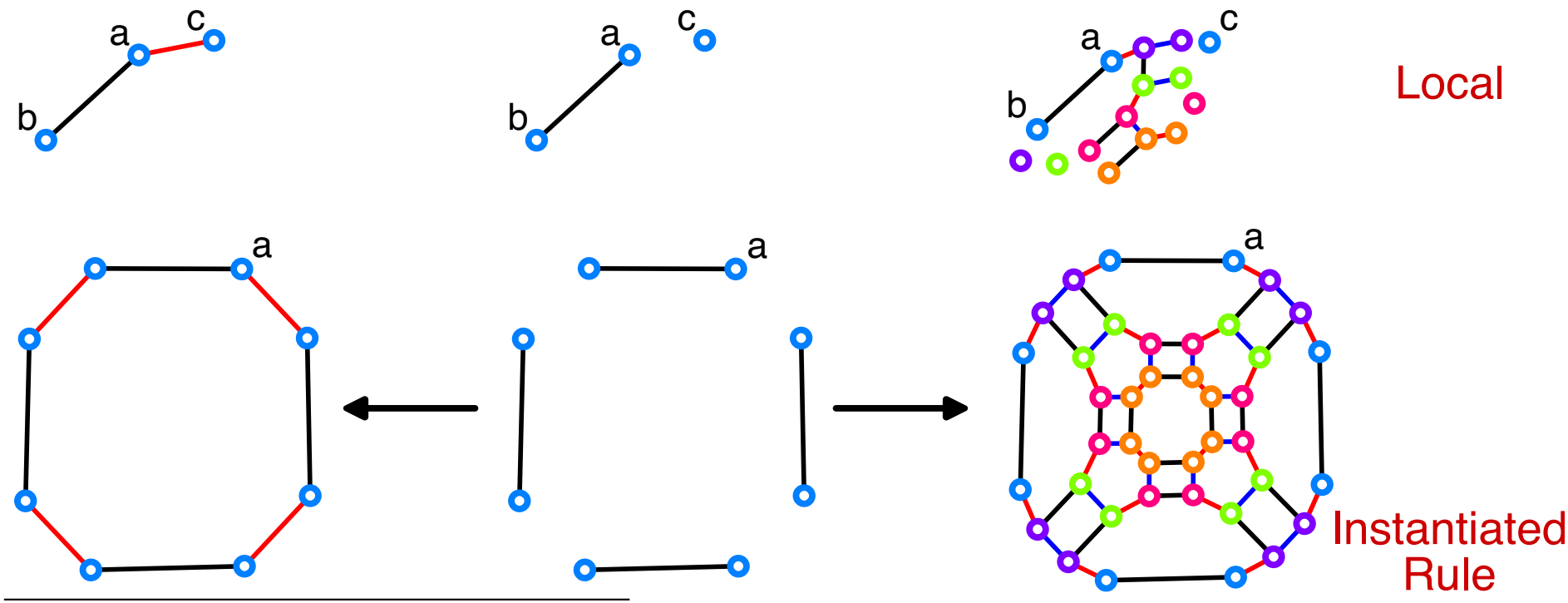
[1] Pascual et al. 2022b.

Orbit Rewriting[1]



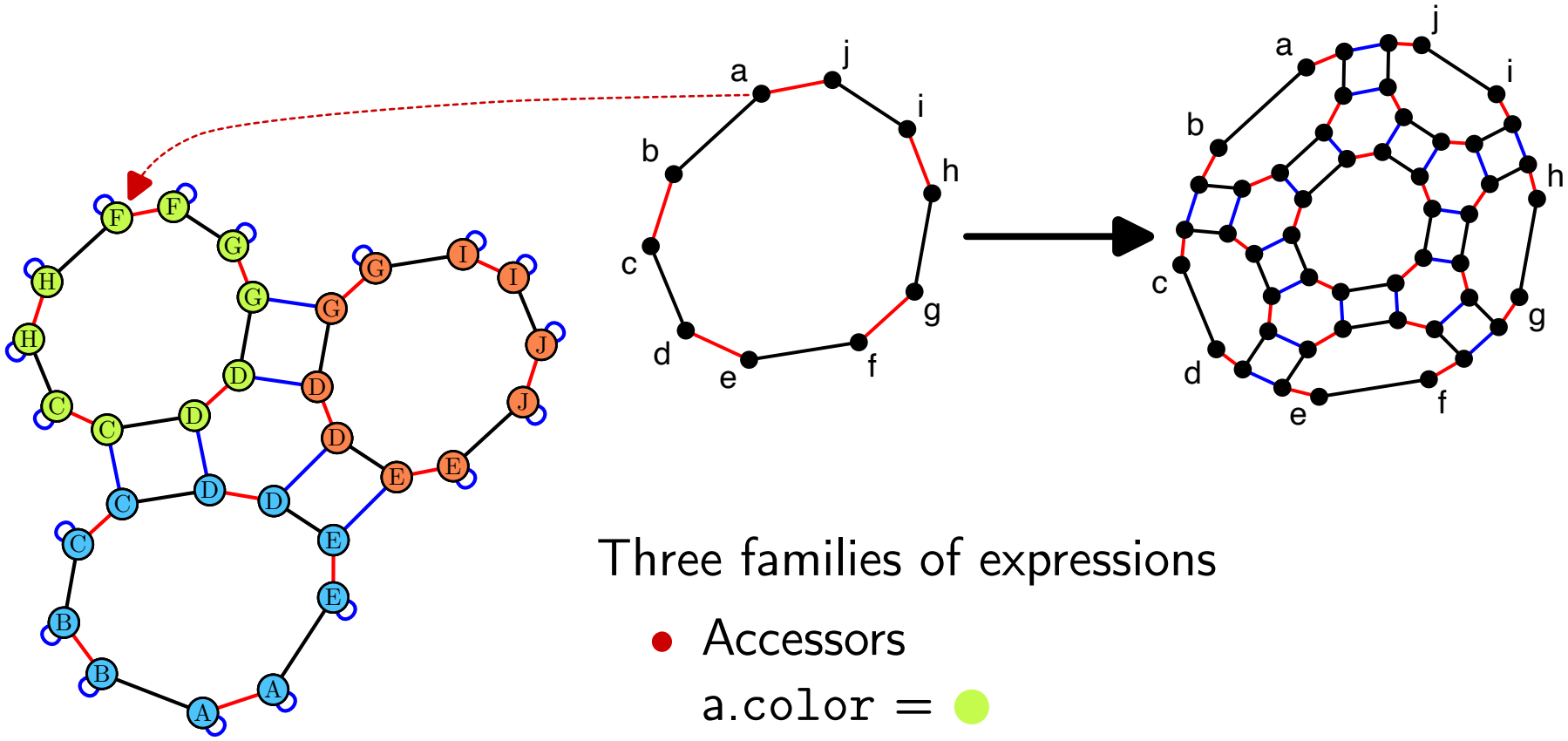


Implicilty Computed



[1]Pascual et al. 2022b.

Embedding Expressions[1]



Three families of expressions

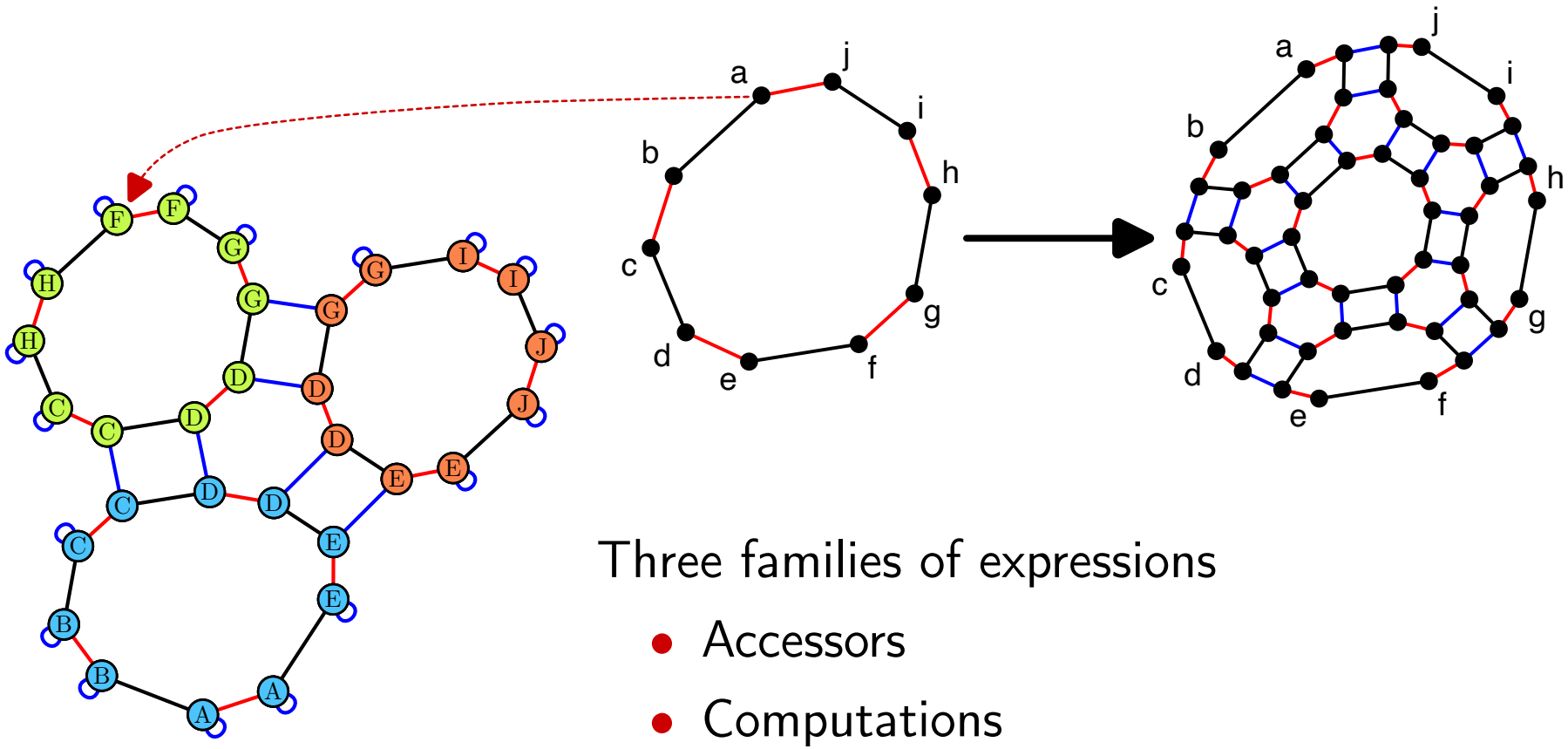
- Accessors

  `a.color` = ●

  `a.position` = F

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

Embedding Expressions[1]



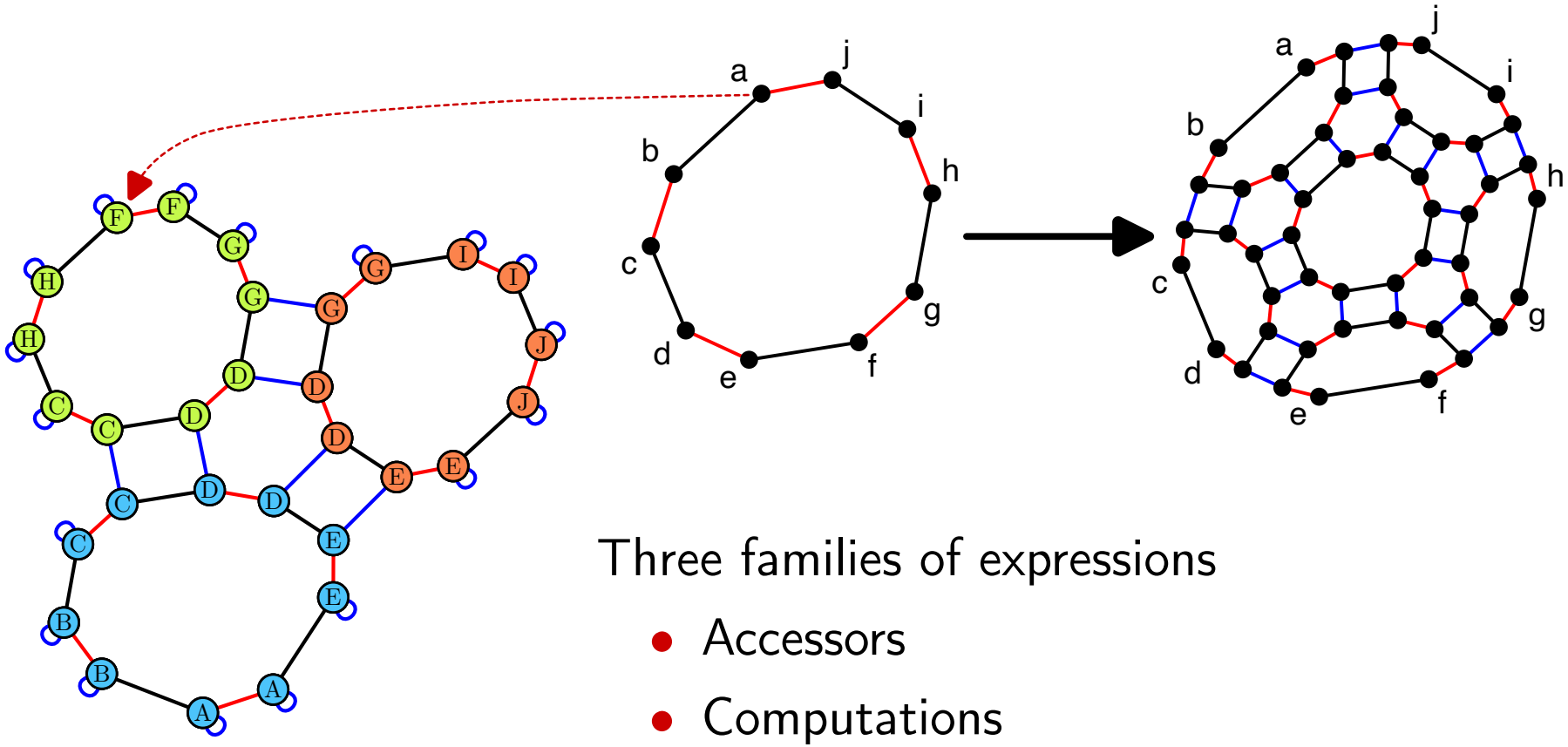
Three families of expressions

- Accessors
- Computations

`middle`({●, ●}) = ●

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

Embedding Expressions[1]

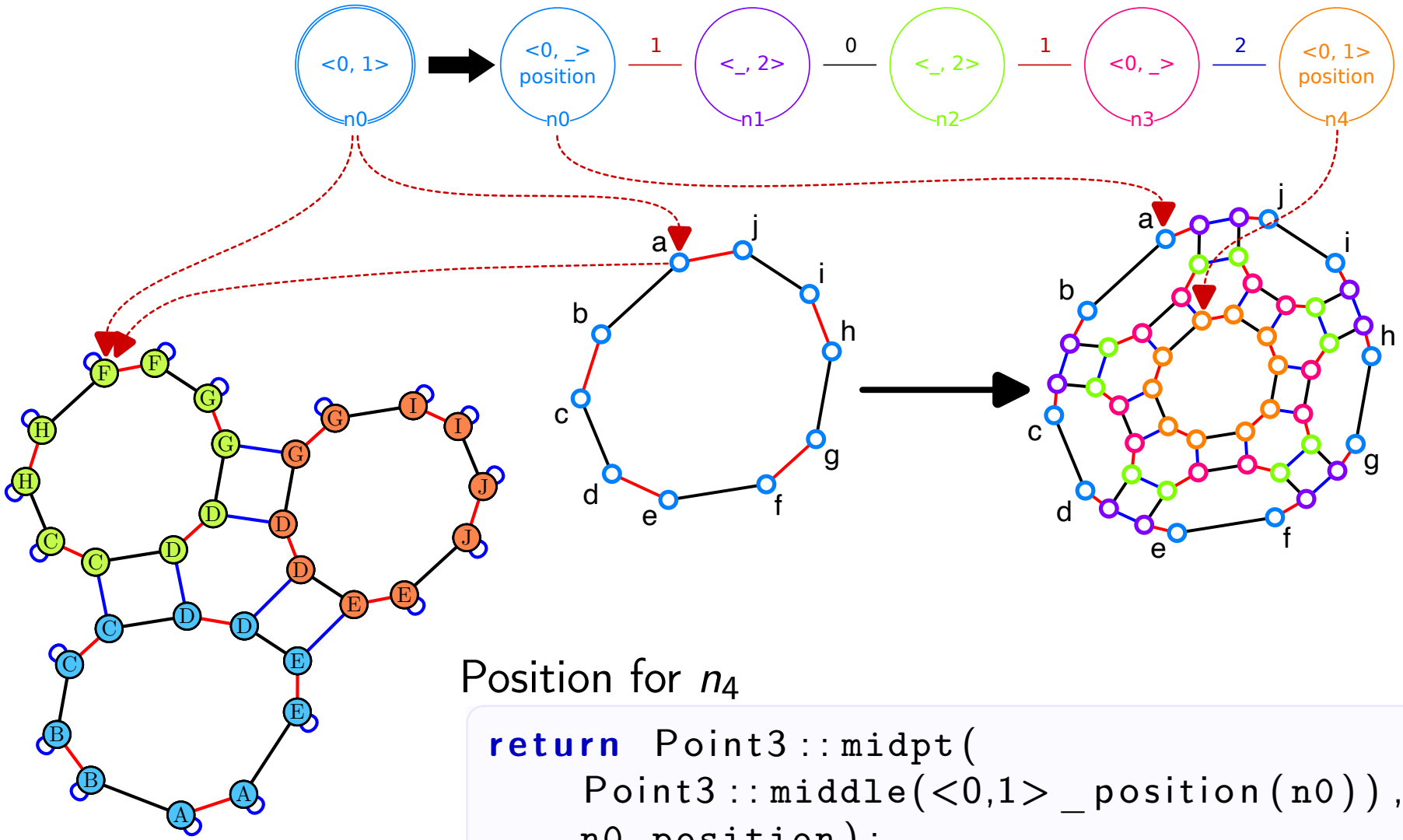


Three families of expressions

- Accessors
- Computations
- Gmap traversals
  $\texttt{position}_{\langle 0,1\rangle}(a) = \{C, D, F, G, H\}$

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

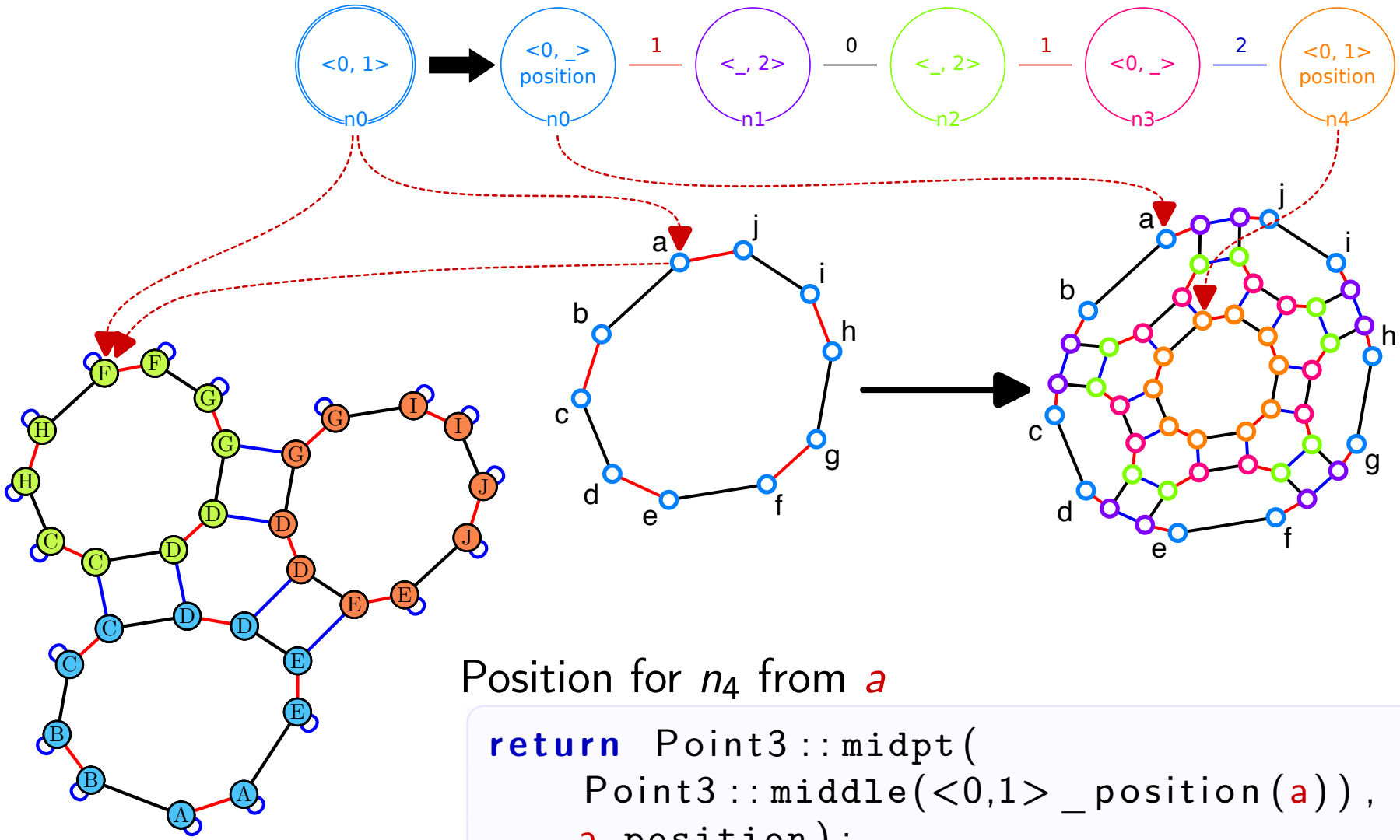
Extension to Rule Schemes



Position for n_4

```
return Point3::midpt(
    Point3::middle(<0,1>_position(n0)),
    n0.position);
```

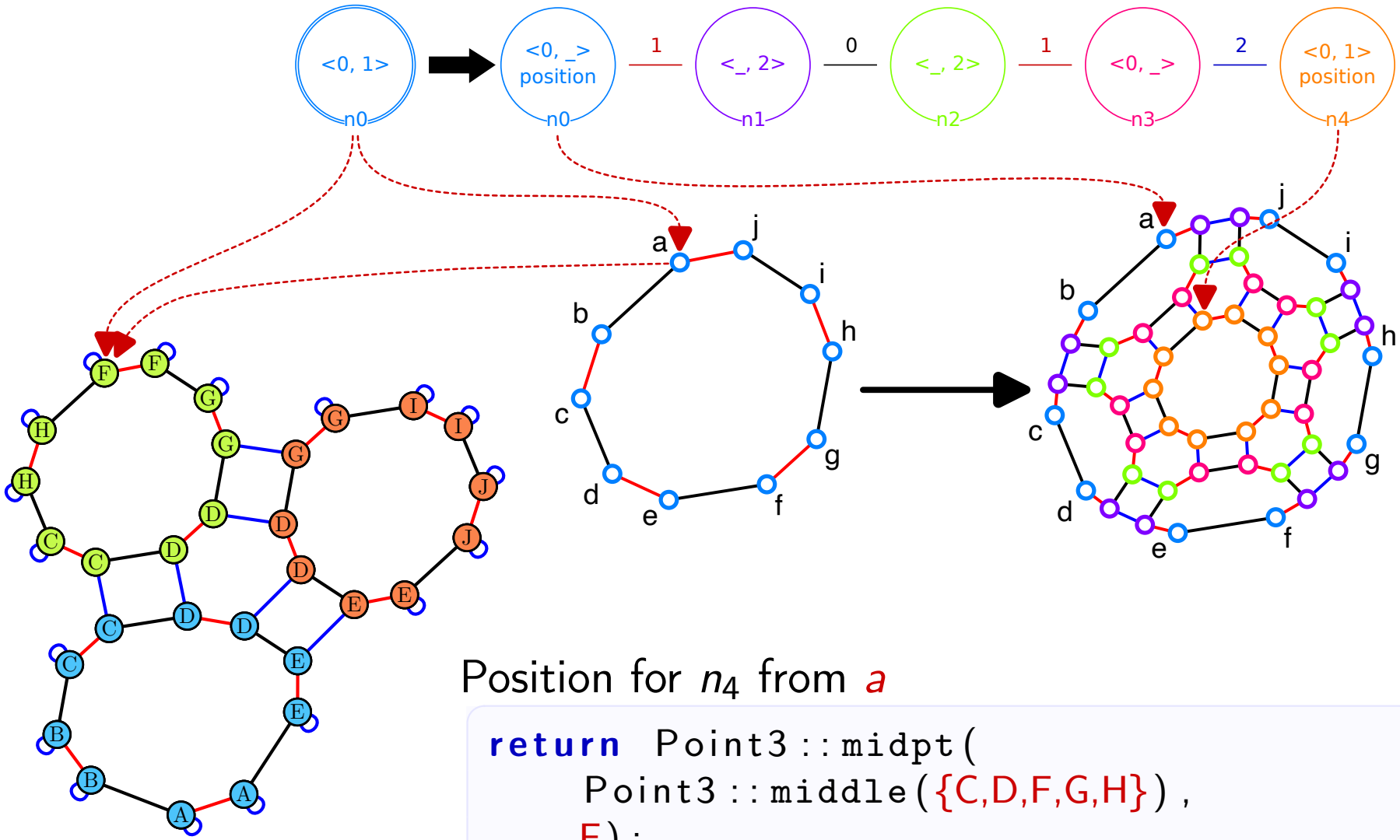
Extension to Rule Schemes



Position for n_4 from *a*

```
return Point3::midpt(
    Point3::middle(<0,1>_position(a)),
    a.position);
```

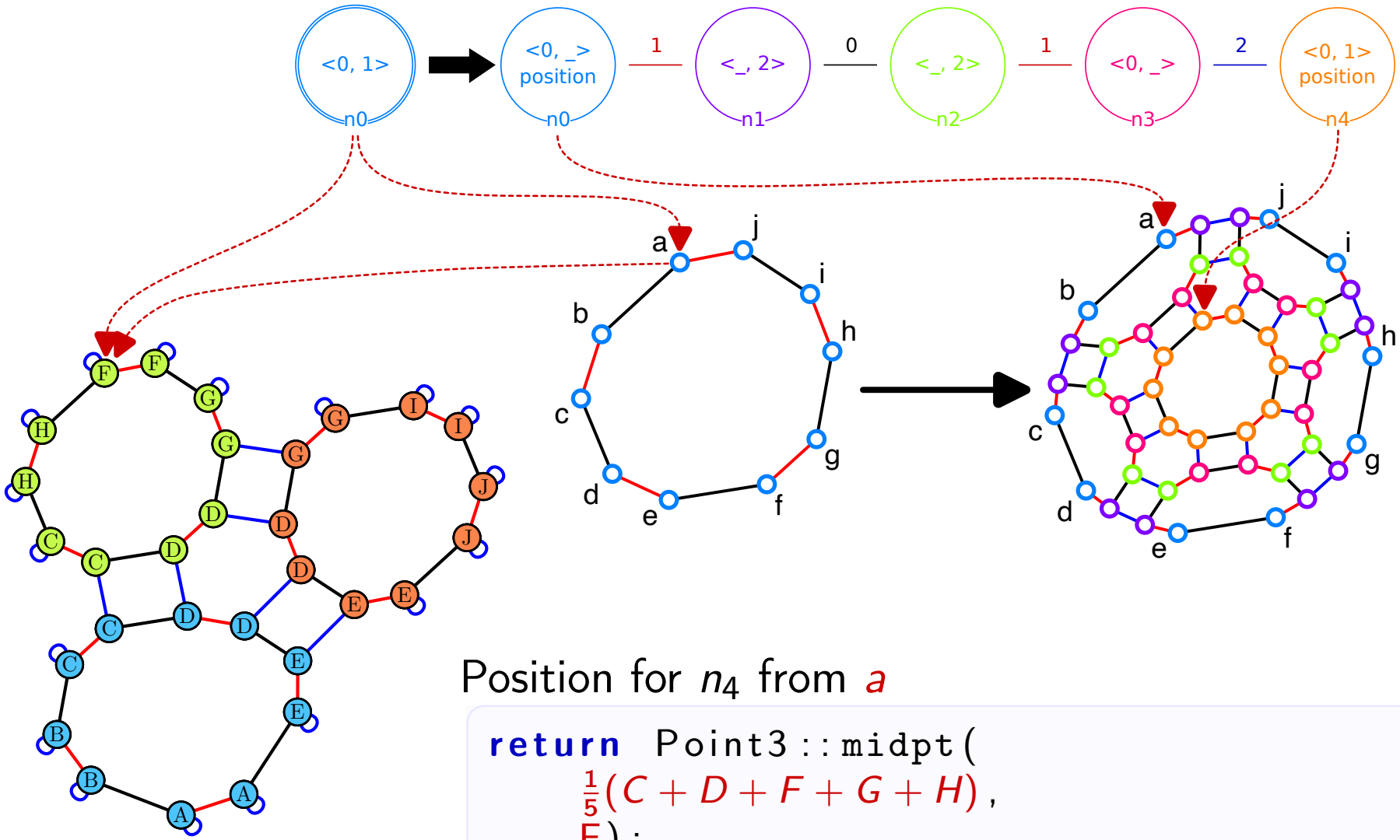
Extension to Rule Schemes



Position for n_4 from *a*

```
return Point3::midpt(
    Point3::middle({C,D,F,G,H}),
    F);
```

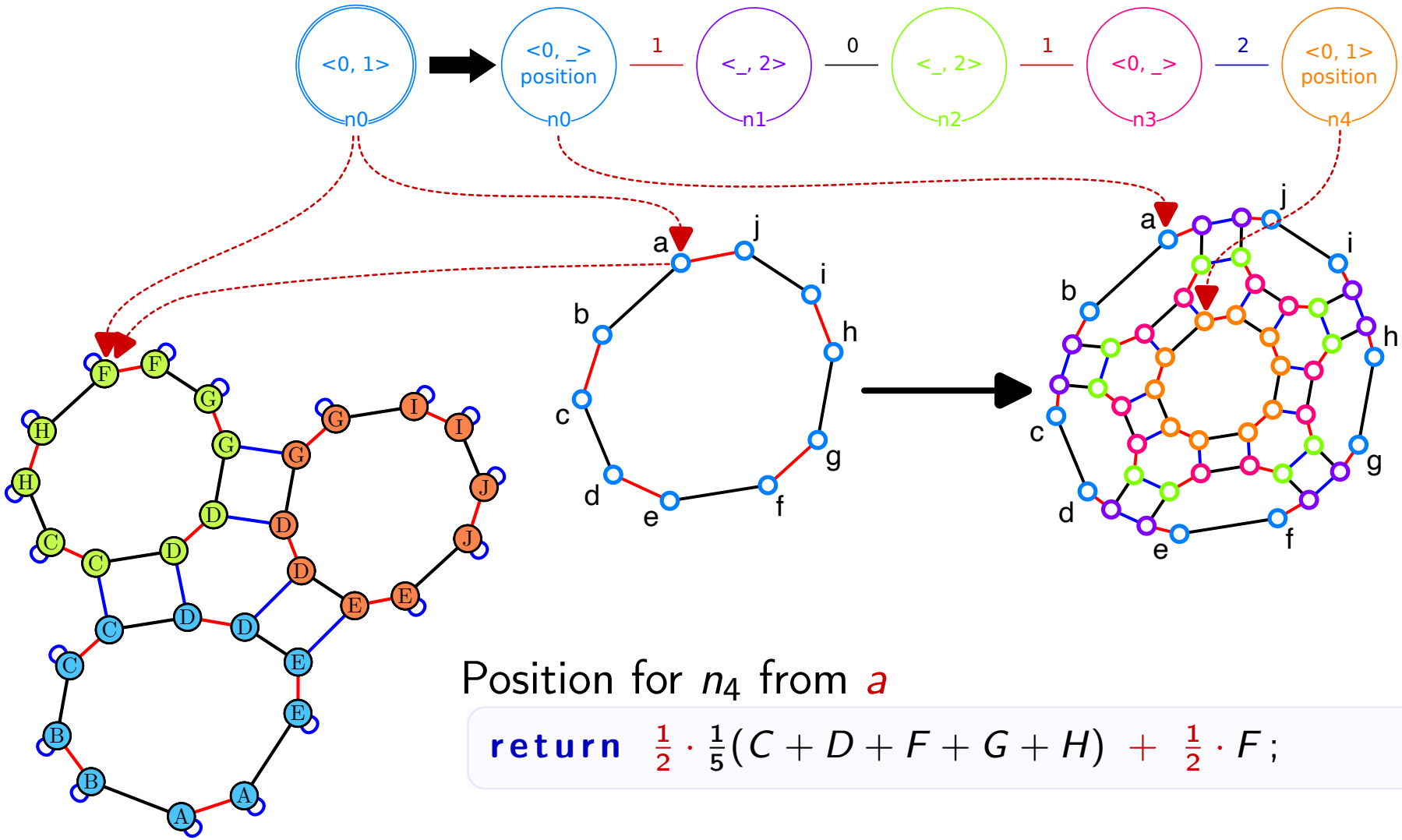
Extension to Rule Schemes



Position for n_4 from a

```
return Point3::midpt(
    1/5(C + D + F + G + H),
    F);
```

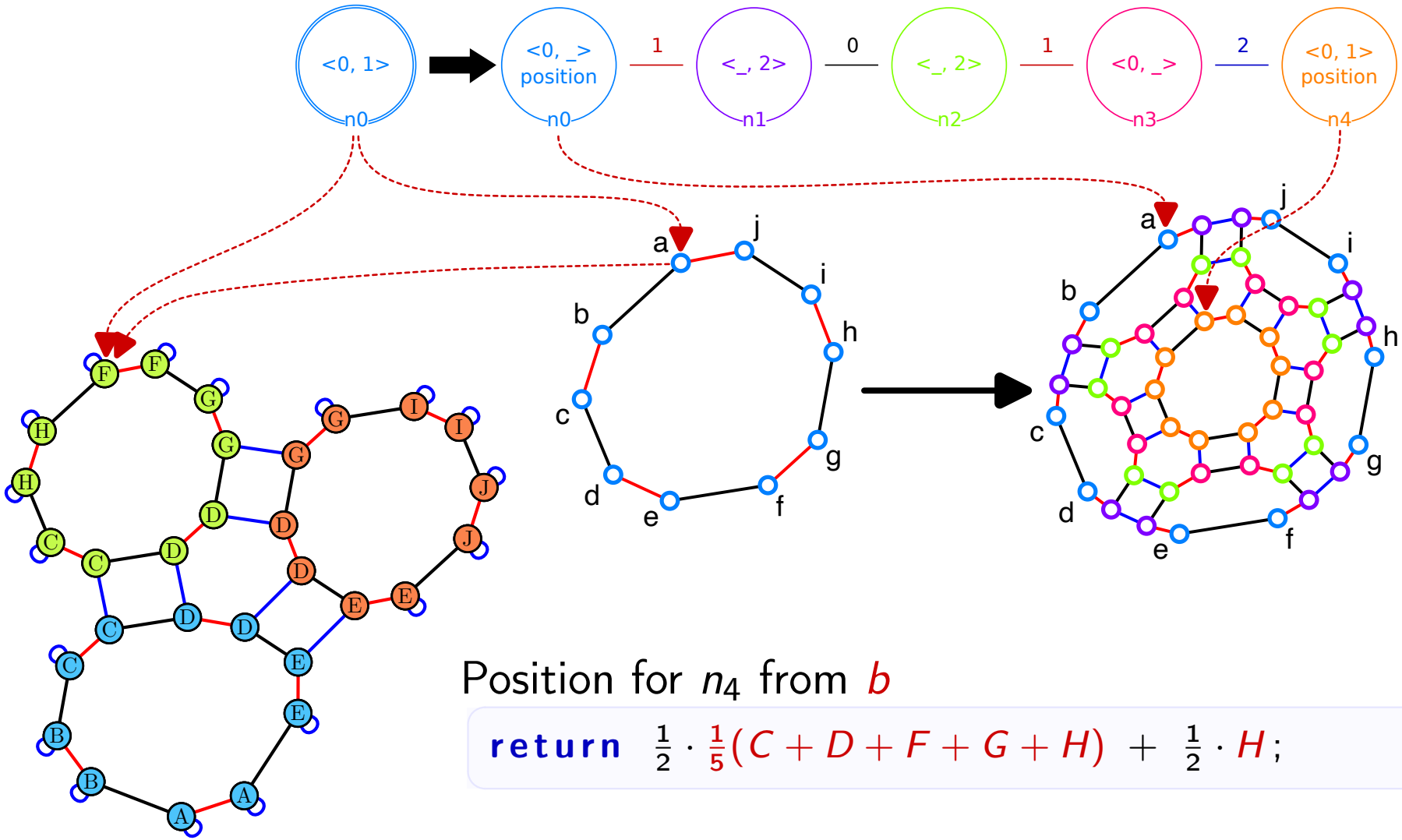
Extension to Rule Schemes



Position for n_4 from a

```
return 1/2 · 1/5 (C + D + F + G + H) + 1/2 · F ;
```

Extension to Rule Schemes

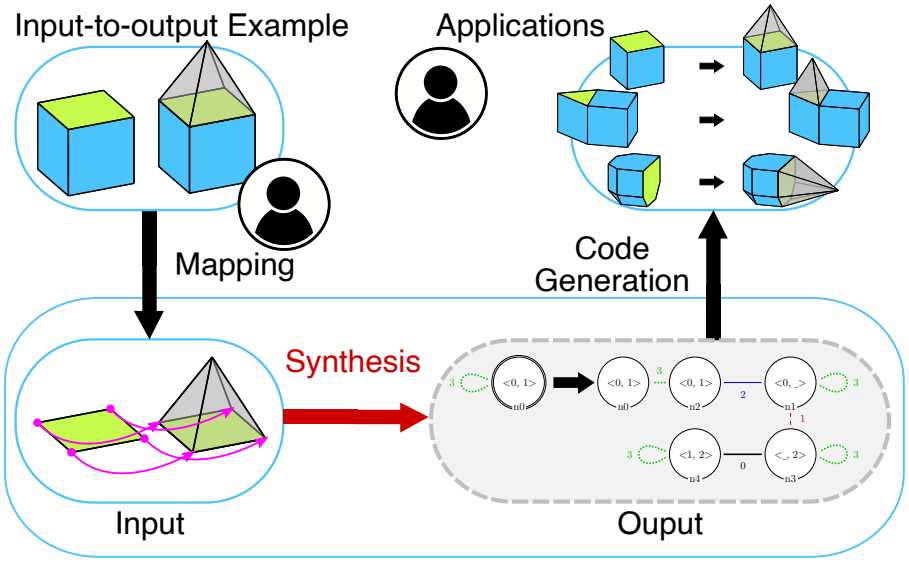


Position for n_4 from b

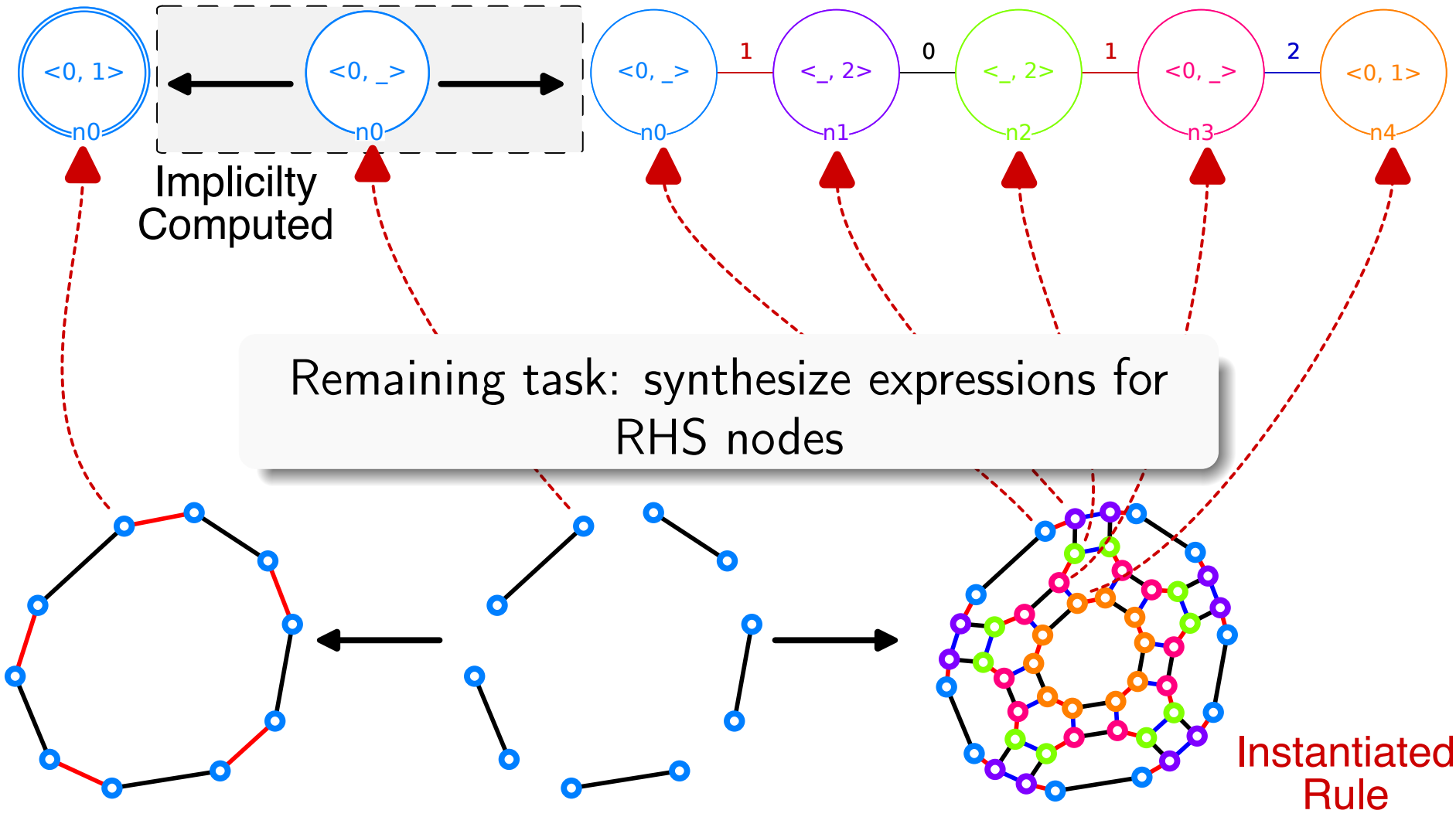
return $\frac{1}{2} \cdot \frac{1}{5}(C + D + F + G + H) + \frac{1}{2} \cdot H$;

Synthesizing Geometric Operations

- How to retrieve the missing embedding expressions?

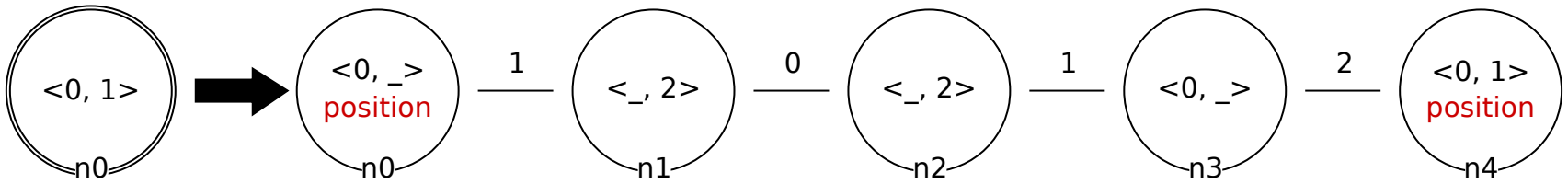


Topological Folding Algorithm[1]



[1] Pascual et al. 2022a.

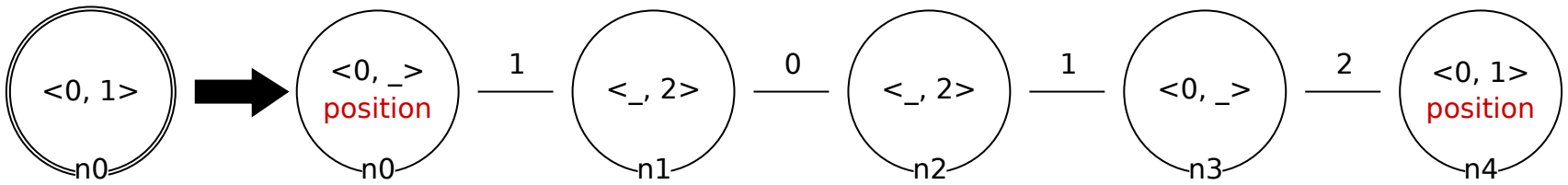
Synthesis as a Syntax-Guided Problem[1]



- Where do candidate expressions come from?
- How do we pick the right one?

[1] Alur et al. 2013.

Synthesis as a Syntax-Guided Problem[1]



- Where do candidate expressions come from?
- How do we pick the right one?

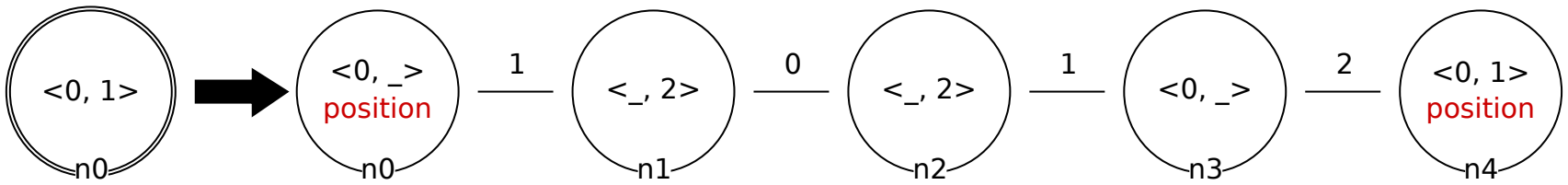
Given

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

Find an expression $e \in L$ such that $\varphi[f/e]$ is valid

[1] Alur et al. 2013.

Synthesis as a Syntax-Guided Problem[1]



- Where do candidate expressions come from?
- How do we pick the right one?

Given

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

Find an expression $e \in L$ such that $\varphi[f/e]$ is valid

What are L and φ ?

[1] Alur et al. 2013.

Overview

Motivation: generalize independently of the input's topology

[1] Gulwani et al. 2012.
[2] Jha et al. 2010.
[3] Solar-Lezama 2013.

Overview

Motivation: generalize independently of the input's topology

Program synthesis answers

- Programming-by-example[1]: φ derived from input-to-output examples
- Components-based synthesis[2]: orbit-dependent built-in functions
- Sketch-based synthesis[3]: filling holes in a program

[1] Gulwani et al. 2012.
[2] Jha et al. 2010.
[3] Solar-Lezama 2013.

Overview

Motivation: generalize independently of the input's topology

Program synthesis answers

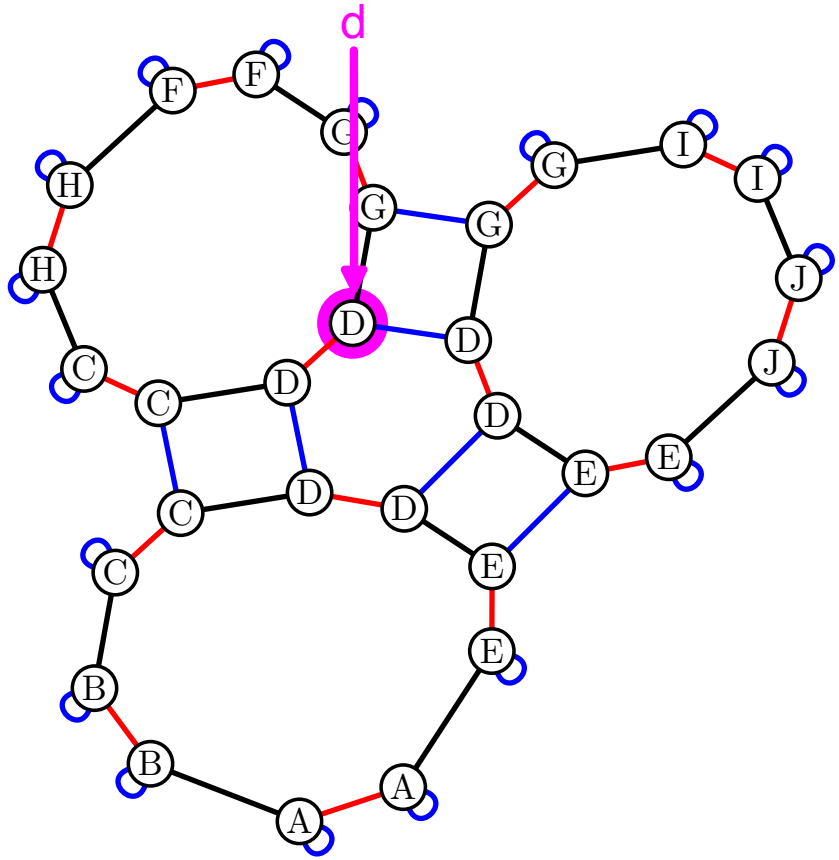
- Programming-by-example[1]: φ derived from input-to-output examples
- Components-based synthesis[2]: orbit-dependent built-in functions
- Sketch-based synthesis[3]: filling holes in a program

Some domain-specific insights

- (Generalized) barycentric coordinates can compute new positions
- Linear arithmetic is a good starting point

[1] Gulwani et al. 2012.
[2] Jha et al. 2010.
[3] Solar-Lezama 2013.

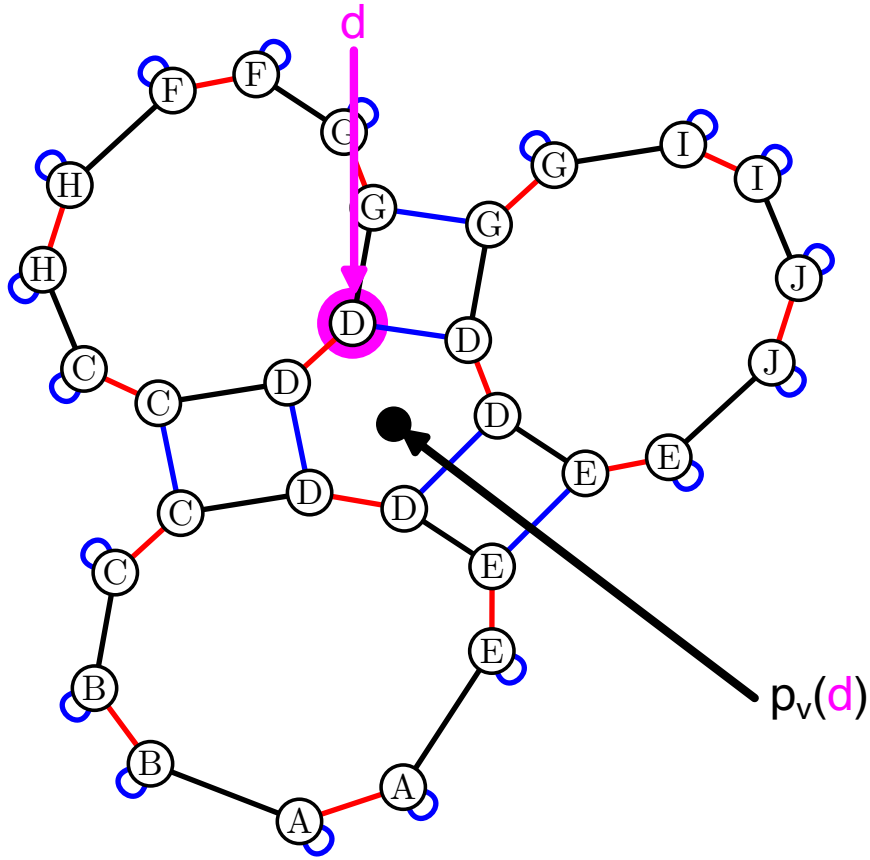
Components: Points of Interest



Components: Points of Interest

with

- p_v : vertex

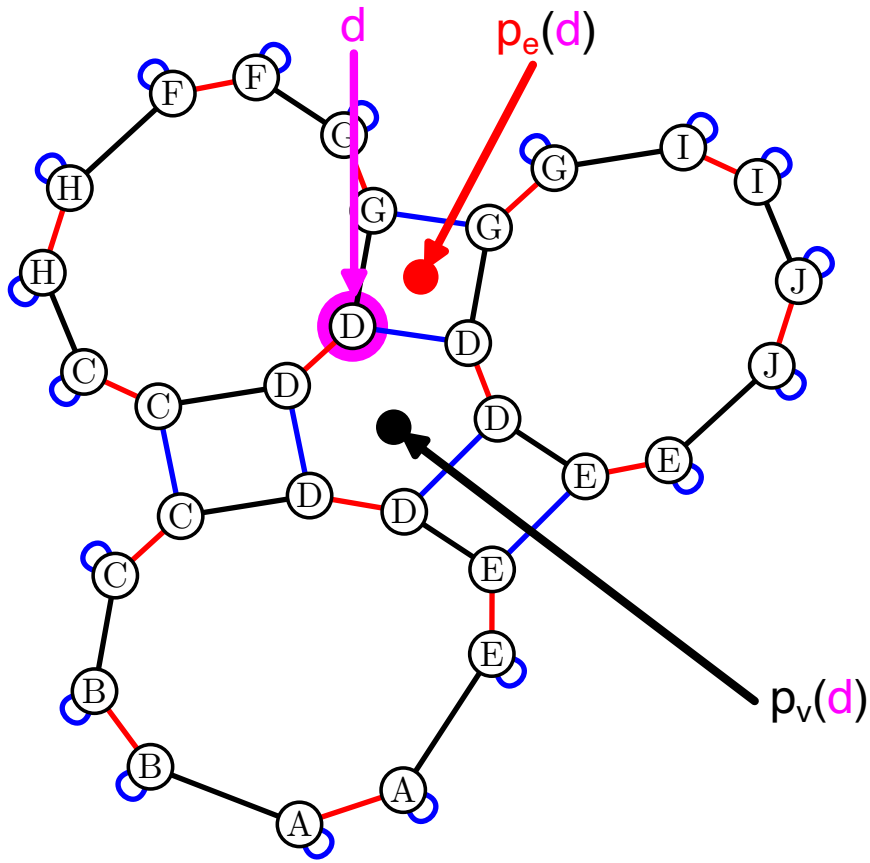


$$p_v = \texttt{middle}(\text{position}_{\langle\rangle}(d))$$

Components: Points of Interest

with

- p_v : vertex
- p_e : edge midpoint

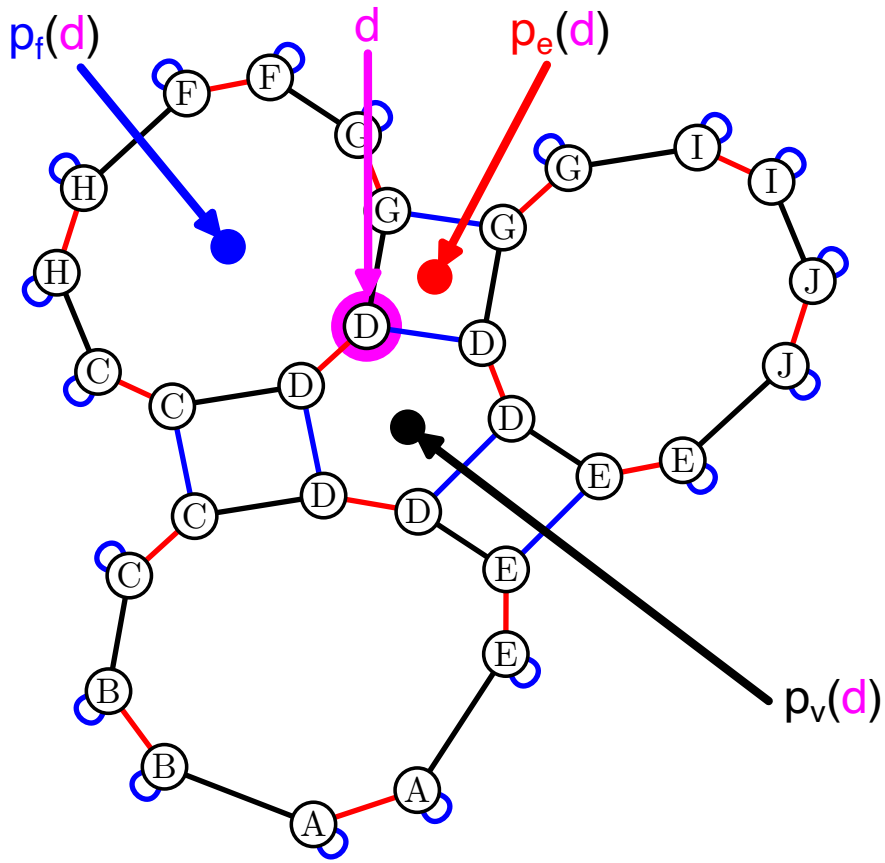


$$p_e = \texttt{middle}(\text{position}_{\langle 0 \rangle}(d))$$

Components: Points of Interest

with

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

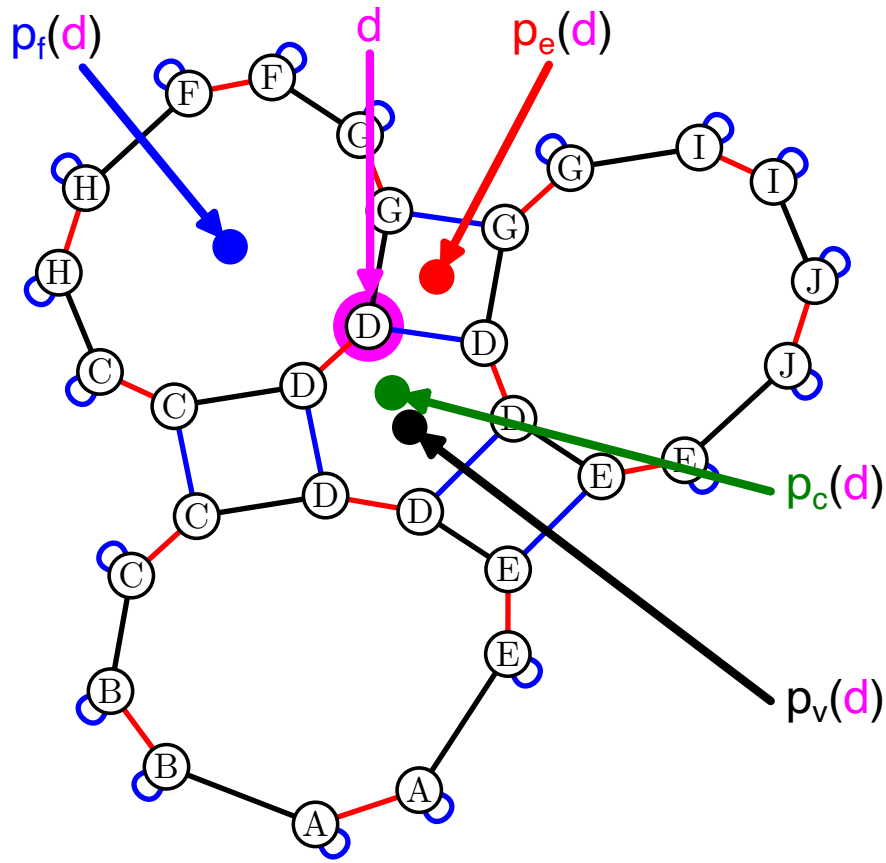


$$p_f = \texttt{middle}(\text{position}_{\langle 0,1 \rangle}(d))$$

Components: Points of Interest

with

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



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

Sketching Geometric Expressions

We fix a generic skeleton

$$\text{position}(n_R) = t + \sum w_{\text{poi}_{\langle o \rangle}(n_L)} \cdot \text{poi}_{\langle o \rangle}(n_L)$$

- $\text{poi}_{\langle o \rangle}(n_L)$: retrieved from LHS nodes
- w: weights to synthesize
- t: translation to synthesize

Sketching Geometric Expressions

We fix a generic skeleton

$$\text{position}(n_R) = t + \sum w_{\text{poi}_{\langle o \rangle}(n_L)} \cdot \text{poi}_{\langle o \rangle}(n_L)$$

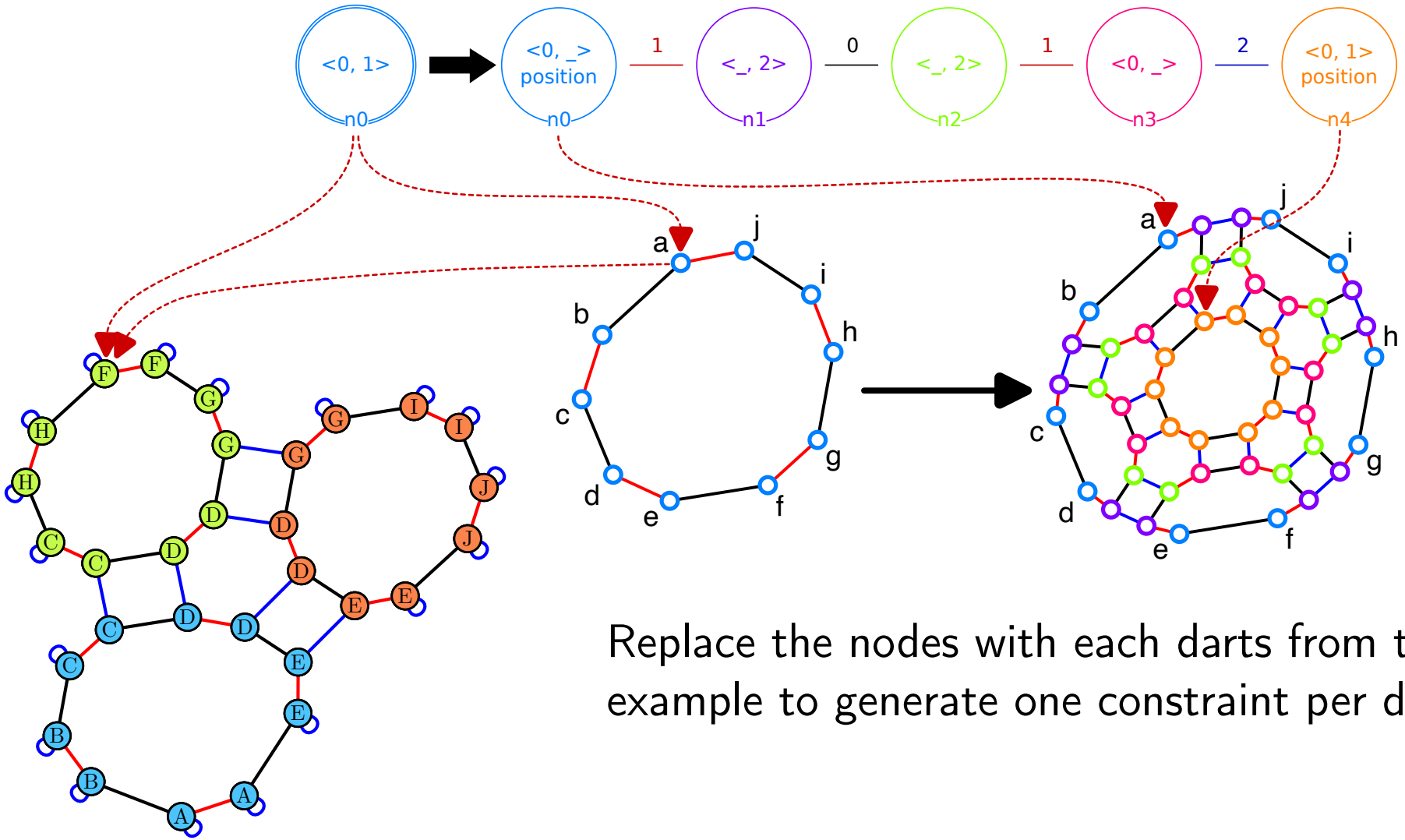
- $\text{poi}_{\langle o \rangle}(n_L)$: retrieved from LHS nodes
- w: weights to synthesize
- t: translation to synthesize

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

Sketch Skeleton (?? indicates a value to be synthesized)

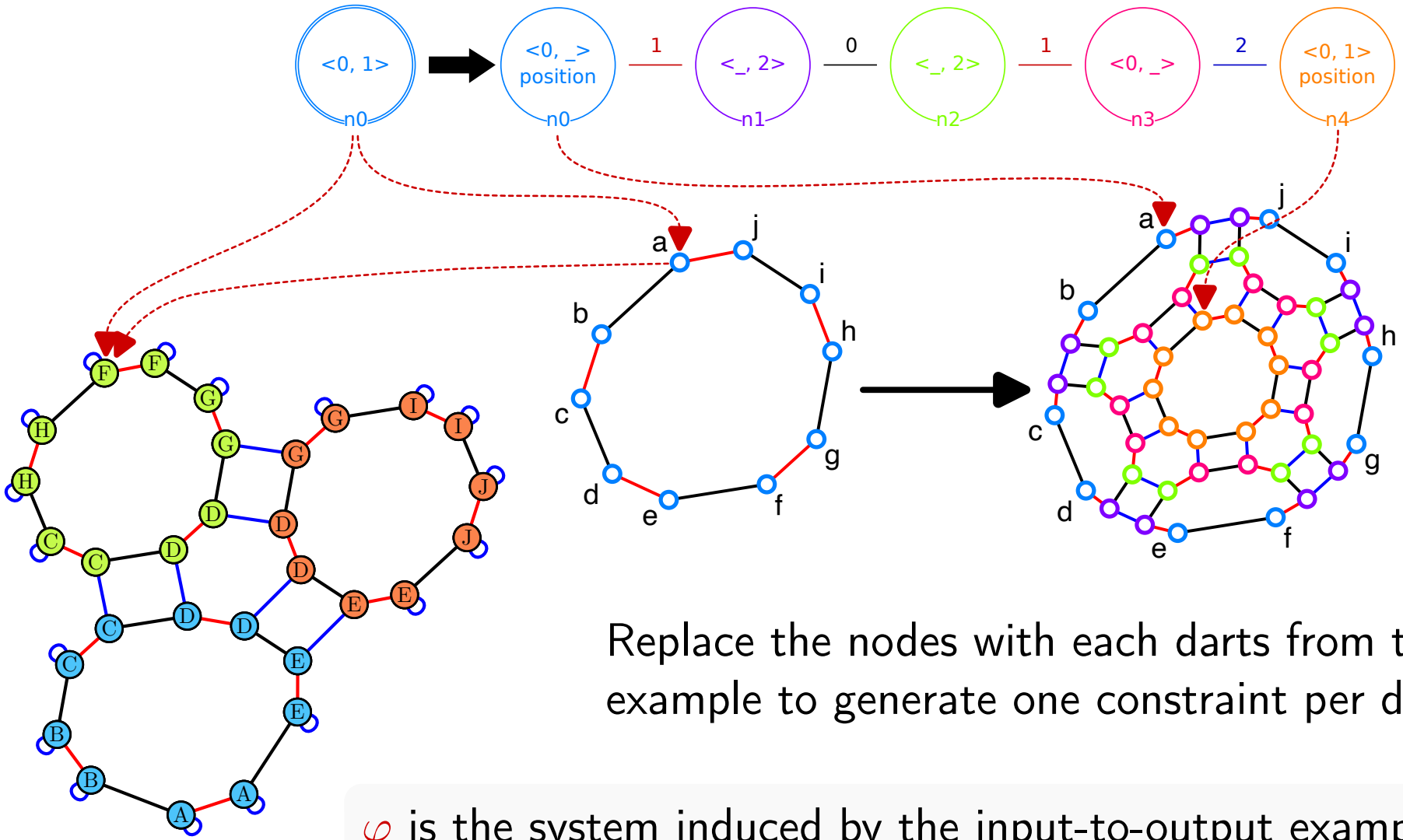
```
// translation
Point3 res = new Point3( ?? , ?? , ?? );

// per #node# in <1,2>-orbit of the left hand side
Point3 pV_#node# = Point3::middle(<1,2>_position( #node# ));
pV_#node#.scaleVect( ?? );
res.addVect(pV_#node#);

// per #node# in <0,2>-orbit of the left hand side
Point3 pE_#node# = Point3::middle(<0,2>_position( #node# ));
pE_#node#.scaleVect( ?? );
res.addVect(pE_#node#);

// per #node# in <0,1>-orbit of the left hand side
Point3 pF_#node# = Point3::middle(<0,1>_position( #node# ));
pF_#node#.scaleVect( ?? );
res.addVect(pF_#node#);

// per #node# in <0,1,2>-orbit of the left hand side
Point3 pC_#node# = Point3::middle(<0,1,2>_position( #node# ));
pC_#node#.scaleVect( ?? );
res.addVect(pC_#node#);

return res;
```

Building the Logical Specification



Replace the nodes with each darts from the example to generate one constraint per dart

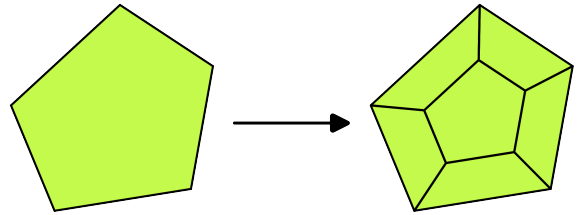
Building the Logical Specification



Replace the nodes with each darts from the example to generate one constraint per dart

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

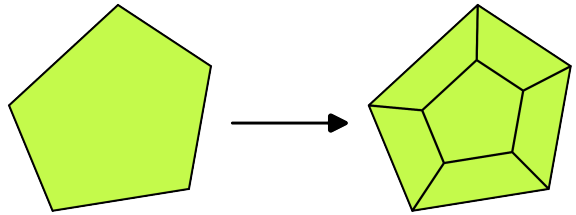
Solving the Flat Extrusion



Symbolic equation

$$\text{position}(n4) = t + w_v \cdot p_v(n0) + w_e \cdot p_e(n0) + w_f \cdot p_f(n0) + w_s \cdot p_s(n0).$$

Solving the Flat Extrusion



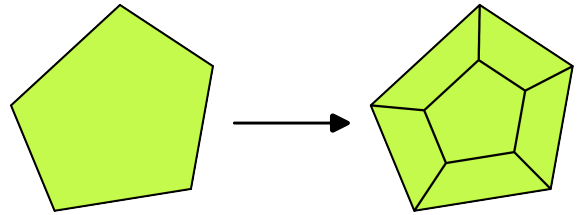
Symbolic equation

$$\text{position}(n4) = t + w_v \cdot p_v(n0) + w_e \cdot p_e(n0) + w_f \cdot p_f(n0) + w_s \cdot p_s(n0).$$

Generated system

$$\begin{cases} 0.449 = tx + w_v \cdot 0.025 + w_e \cdot 0.261 + w_f \cdot 0.874 + w_s \cdot 0.874 \\ 0.935 = ty + w_v \cdot 0.927 + w_e \cdot 0.567 + w_f \cdot 0.943 + w_s \cdot 0.943 \\ 0.685 = tx + w_v \cdot 0.496 + w_e \cdot 0.261 + w_f \cdot 0.874 + w_s \cdot 0.874 \\ 0.575 = ty + w_v \cdot 0.208 + w_e \cdot 0.567 + w_f \cdot 0.943 + w_s \cdot 0.943 \\ \quad \vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots \end{cases}$$

Solving the Flat Extrusion



Symbolic equation

$$\text{position}(n4) = t + w_v \cdot p_v(n0) + w_e \cdot p_e(n0) + w_f \cdot p_f(n0) + w_s \cdot p_s(n0).$$

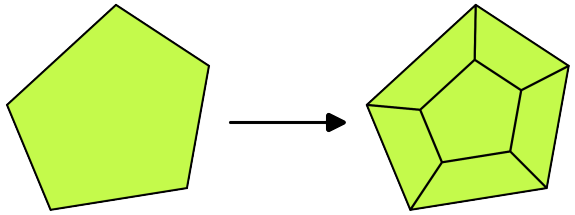
Generated system

$$\begin{cases} 0.449 = tx + w_v \cdot 0.025 + w_e \cdot 0.261 + w_f \cdot 0.874 + w_s \cdot 0.874 \\ 0.935 = ty + w_v \cdot 0.927 + w_e \cdot 0.567 + w_f \cdot 0.943 + w_s \cdot 0.943 \\ 0.685 = tx + w_v \cdot 0.496 + w_e \cdot 0.261 + w_f \cdot 0.874 + w_s \cdot 0.874 \\ 0.575 = ty + w_v \cdot 0.208 + w_e \cdot 0.567 + w_f \cdot 0.943 + w_s \cdot 0.943 \\ \quad \vdots \qquad\qquad\qquad\qquad\qquad \vdots \end{cases}$$

Delegated to a solver (Z3 with QF FP or Or-Tools with GLOP)

- $w_v = 0.5$
- $w_e = 0.0$
- $w_f = 0.5$
- $w_s = 0.0$
- $t = (0.0, 0.0)$

Synthesized Expression (Flat Extrusion)

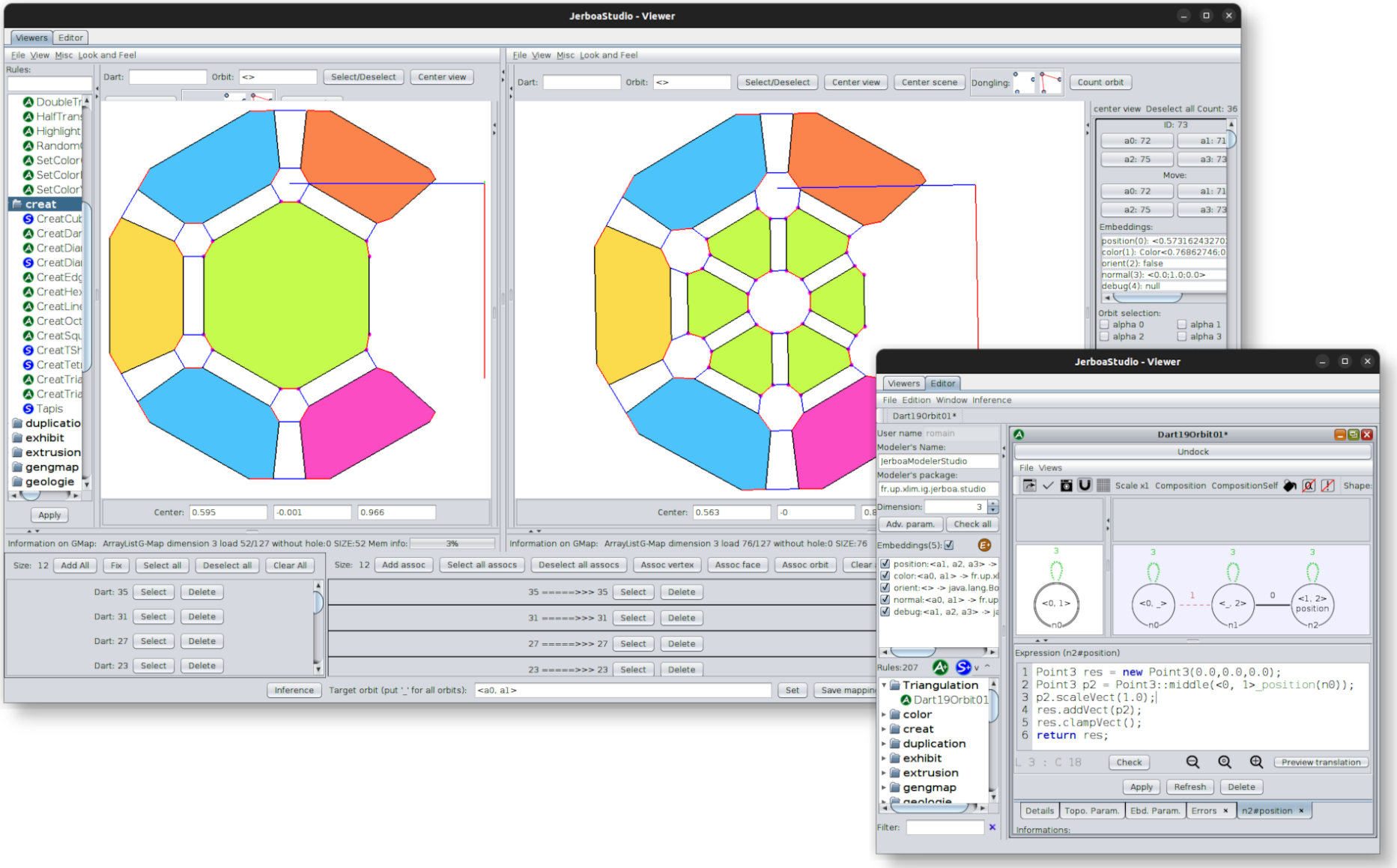


```
// no translation
Point3 res = new Point3(0.0, 0.0, 0.0);

// vertex
Point3 p0 = Point3::middle(<1,2>_position(n0));
p0.scaleVect(0.5);
res.addVect(p0);

// face
Point3 p2 = Point3::middle(<0,1>_position(n0));
p2.scaleVect(0.5);
res.addVect(p2);

return res;
```

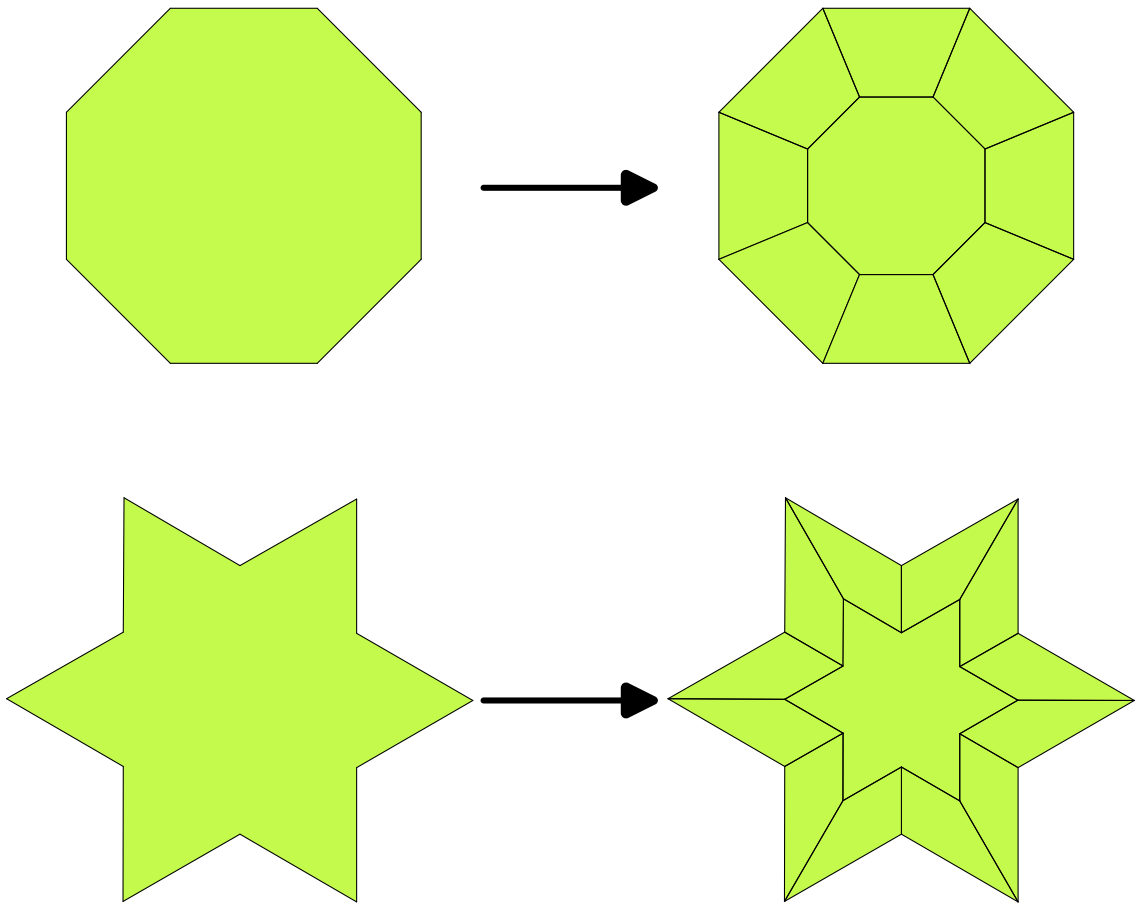
JerboaStudio

- Implementation and Evaluation

JerboaStudio (`https://gitlab.com/jerboateam/jerboa-studio`)

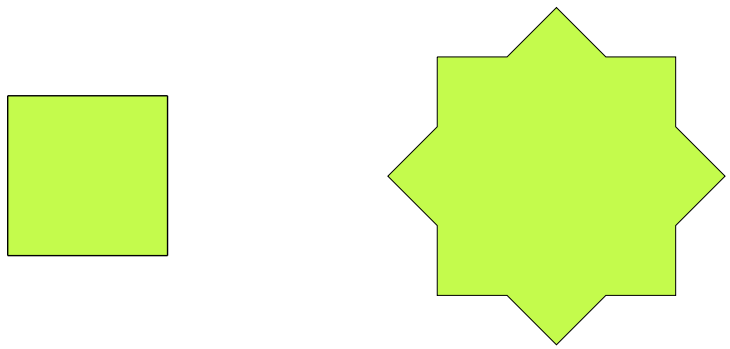


Applying the Synthesized Operation

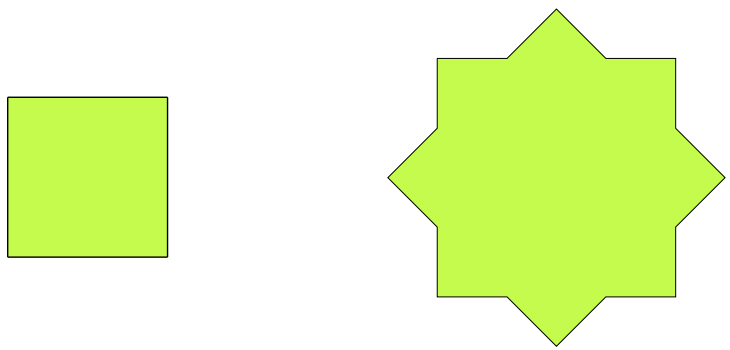


Limits

Von Koch's snowflake generated by L-systems

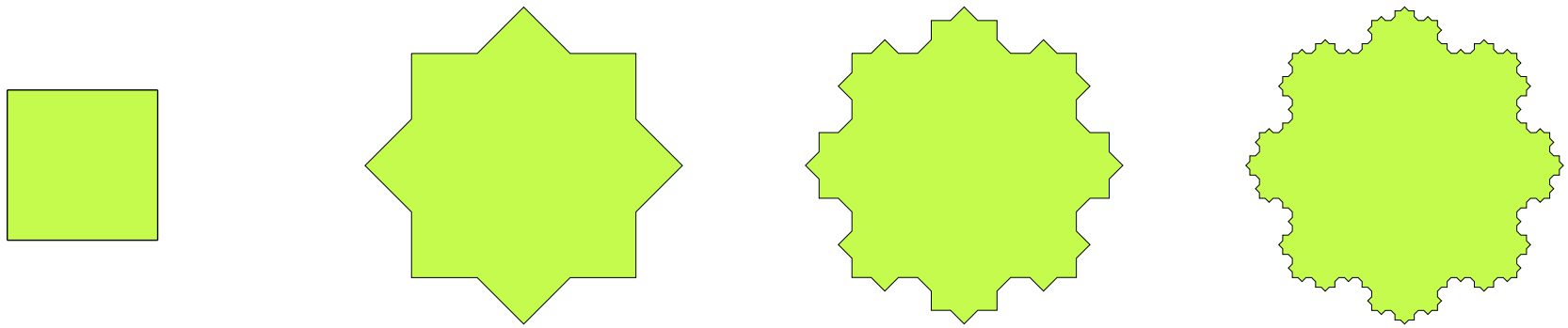


Synthesized

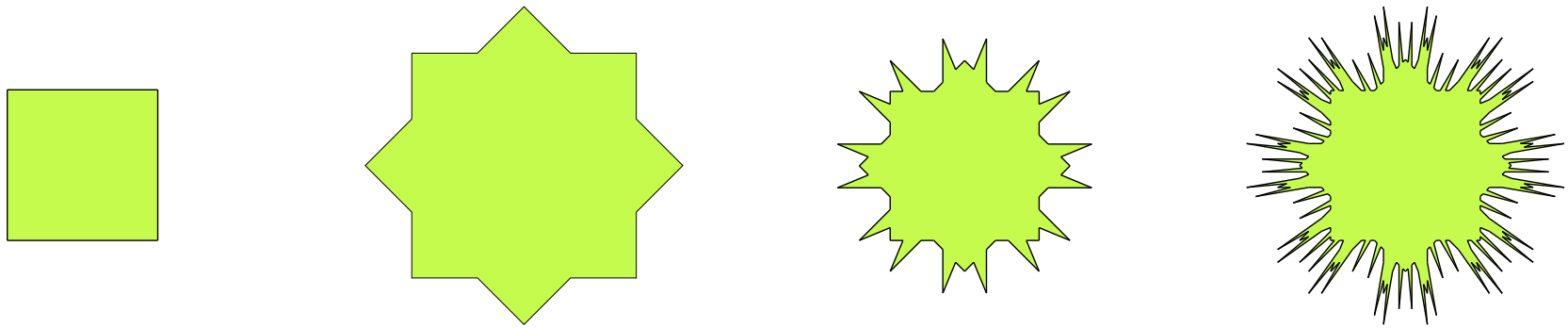


Limits

Von Koch's snowflake generated by L-systems



Synthesized



Program Synthesis for Geometric Modeling

L φ

Program Synthesis for Geometric Modeling

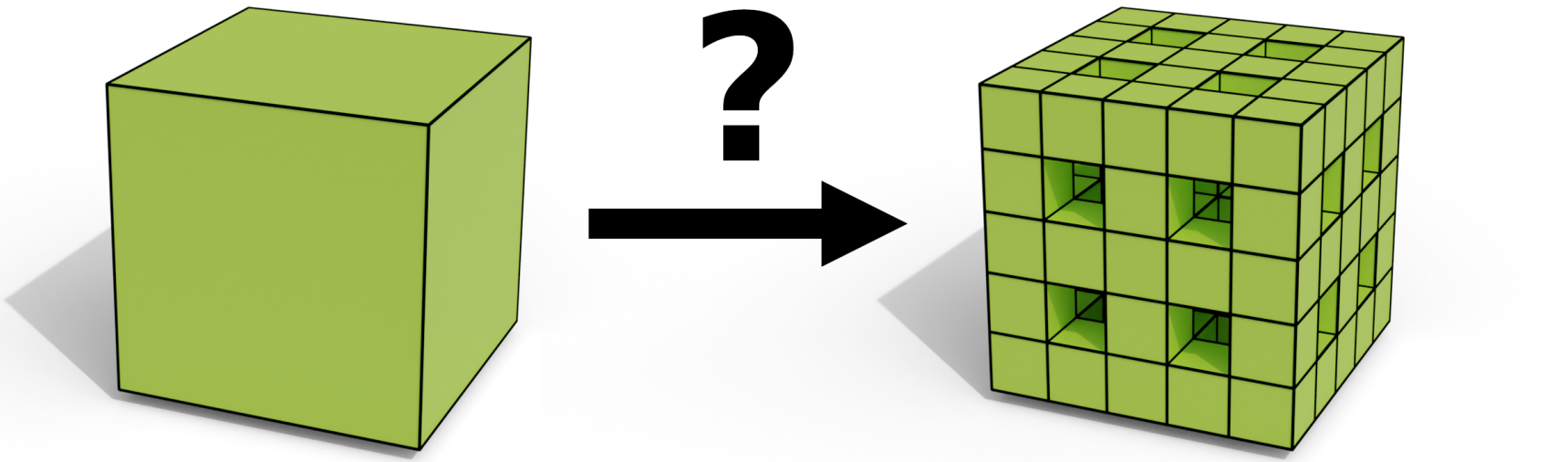
	L	φ
Rule level	Rule scheme	Instantiated rule

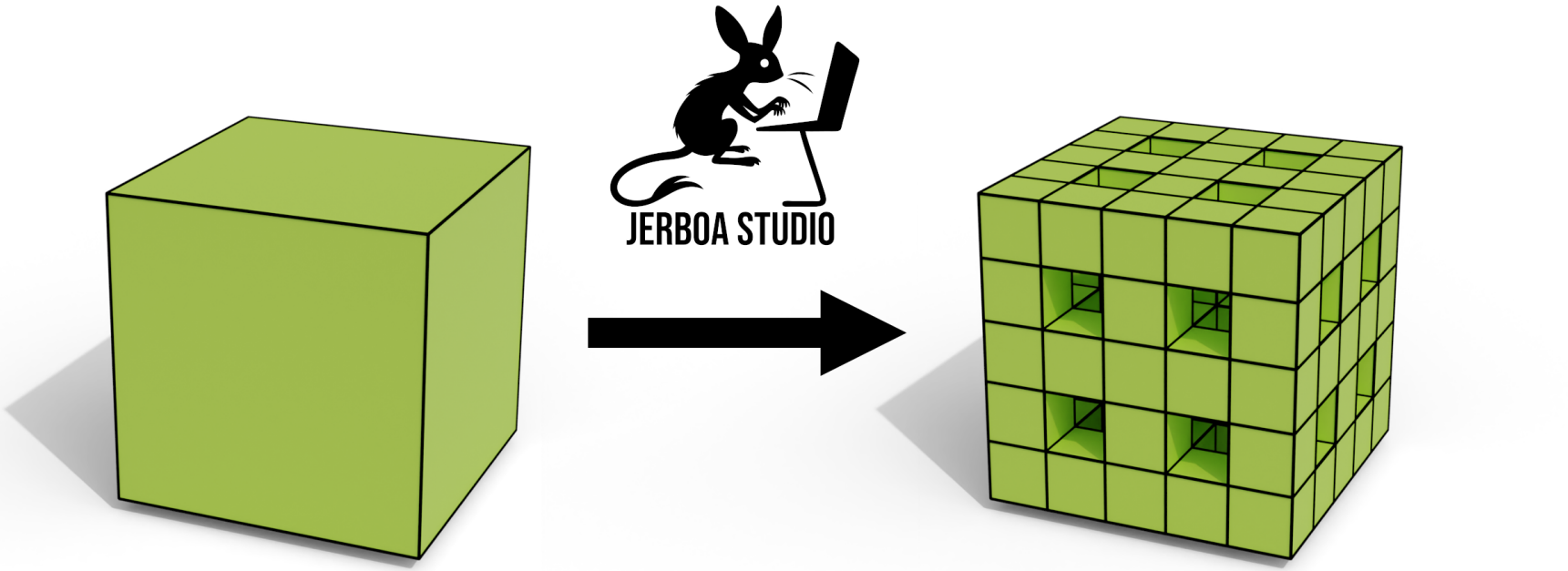
Program Synthesis for Geometric Modeling

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

Program Synthesis for Geometric Modeling

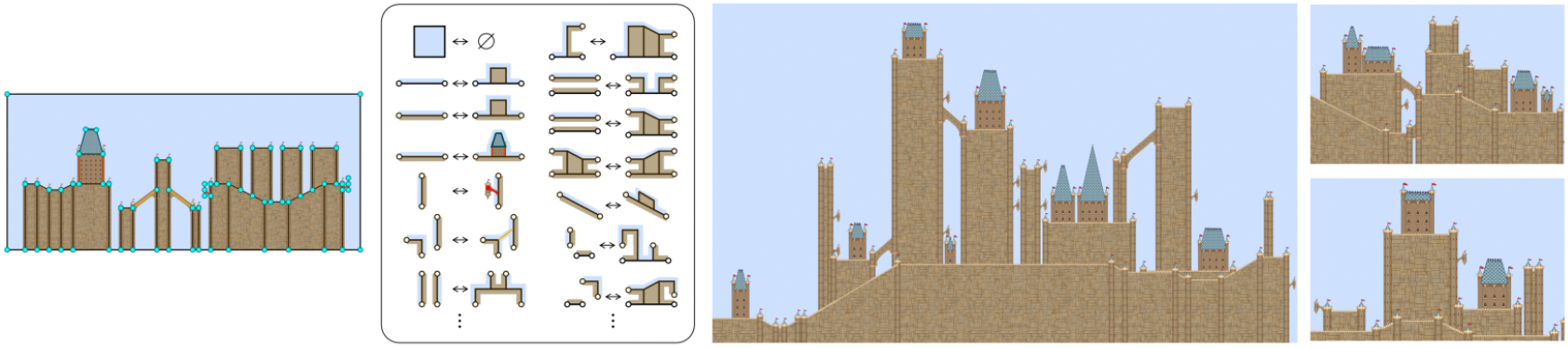
	L	φ
Rule level	Rule scheme	Instantiated rule
Corresponds to	Affine combinations of points of interest	Concrete system derived from the example
Extend with	• other points of interest • other computations	• multi-examples • counter-examples




JERBOA STUDIO

Synthesis in Geometric Modeling

1. Object generation:
   - 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).

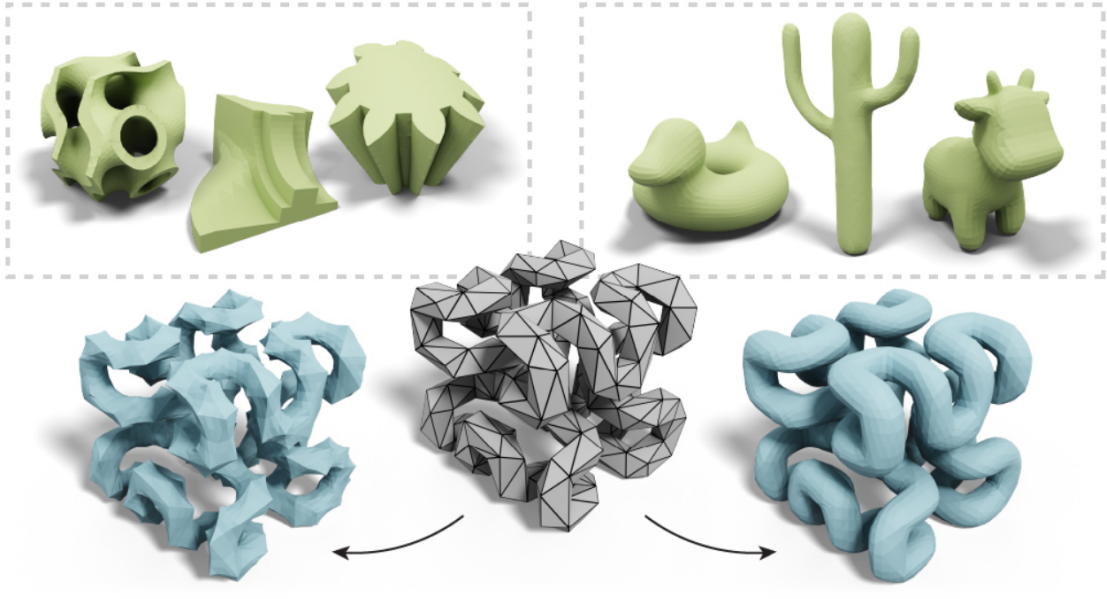


(a) Example Shape (b) Graph Grammar (c) Generated Shapes

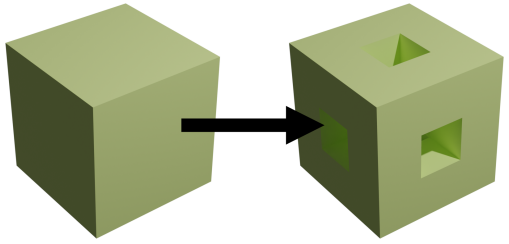
[1] Wu et al. 2014; Emilien et al. 2015.
[2] Santos et al. 2009; Šťava et al. 2010; Guo et al. 2020.
[3] Sharma et al. 2018; Kania et al. 2020; Xu et al. 2021.

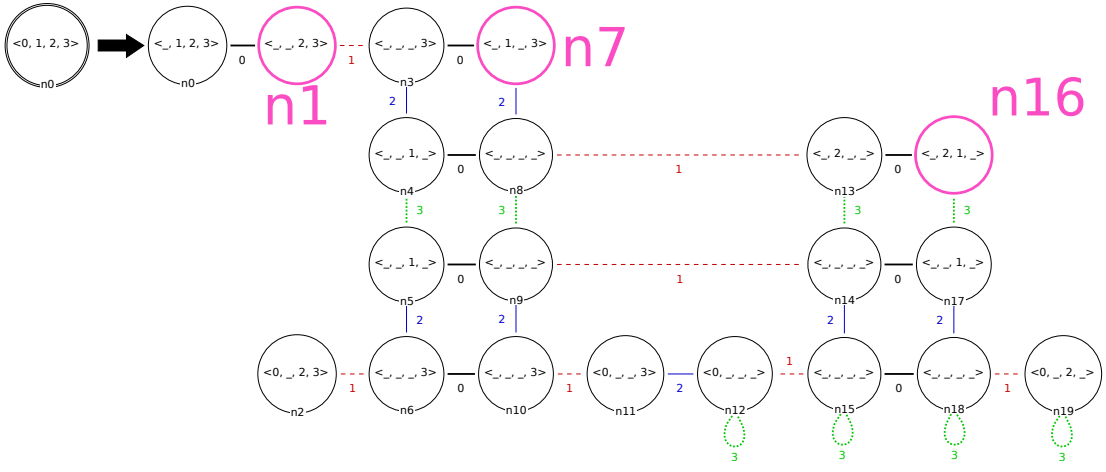
Synthesis in Geometric Modeling

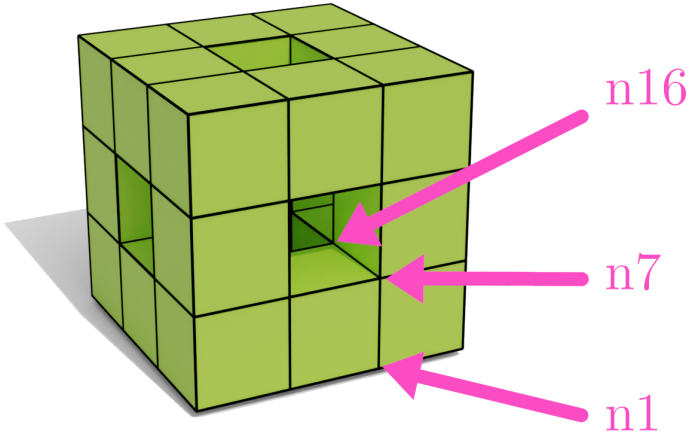
1. Object generation
2. Pure geometry:
   - Retrieve non-linear weights of a Loop-based subdivision scheme for mesh refinement. Illustration from (Liu et al. 2020).



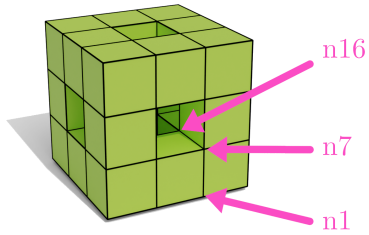
Menger Sponge







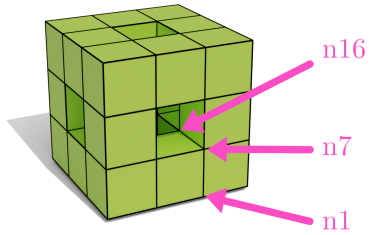
Menger Sponge



Node *n7*

```
Point3 res = new Point3(0.0,0.0,0.0);
Point3 p0 = Point3::middle(<> _position(n0));
p0.scale(0.3333331346511184);
res.addVect(p0);
Point3 p2 = Point3::middle(<0,1> _position(n0));
p2.scale(0.6666668653488816);
res.addVect(p2);
return res;
```

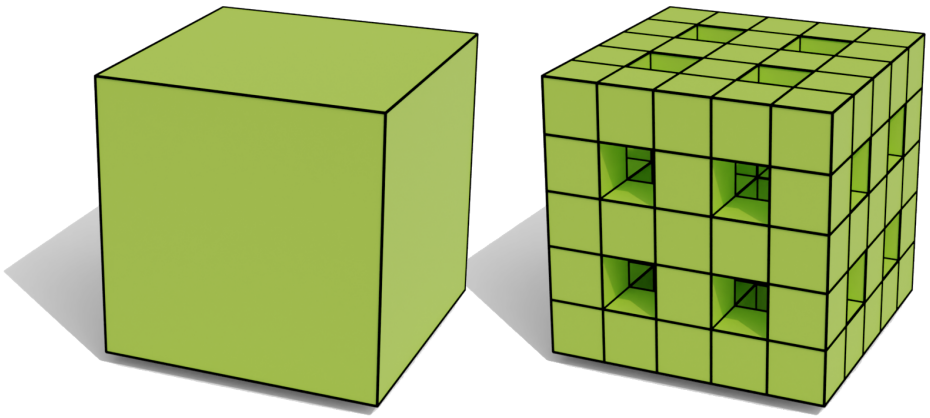
Menger Sponge



Node *n*16

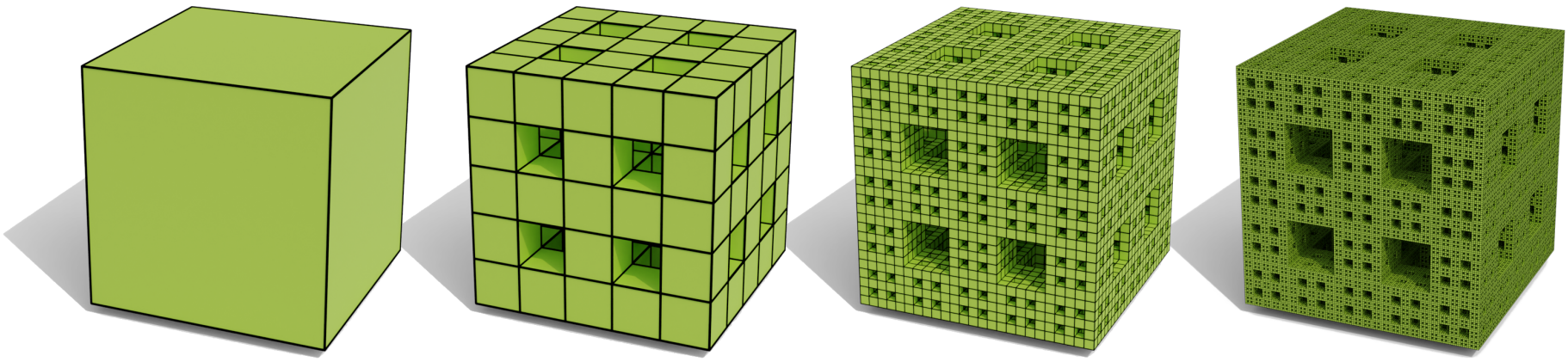
```
Point3 res = new Point3(0.0,0.0,0.0);
Point3 p0 = Point3::middle(<> _position(n0));
p0.scale(0.3333331346511184);
res.addVect(p0);
Point3 p3 = Point3::middle(<0,1,2> _position(n0));
p3.scale(0.6666666865348816);
res.addVect(p3);
return res;
```

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



[1] Richaume et al. 2019.

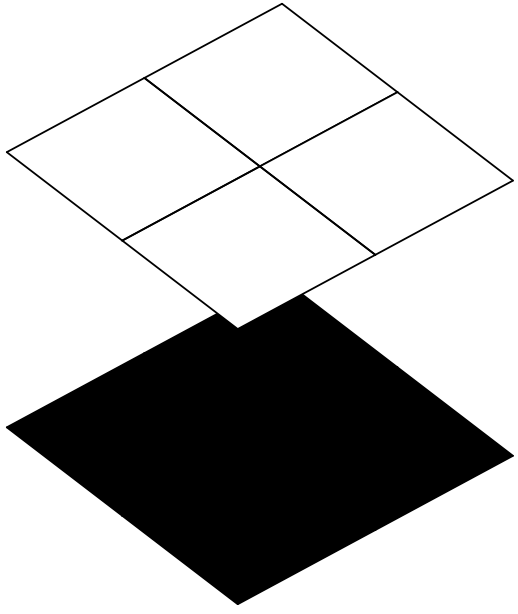
$(2, 2, 2)$-Menger Polycube[1]



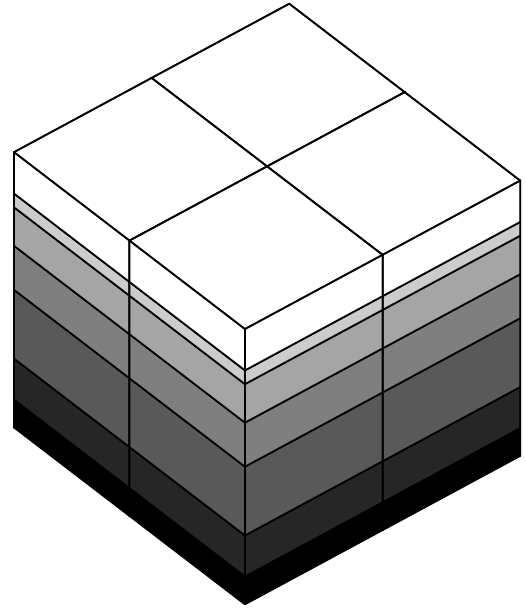
[1] Richaume et al. 2019.

Geology inspired

Before



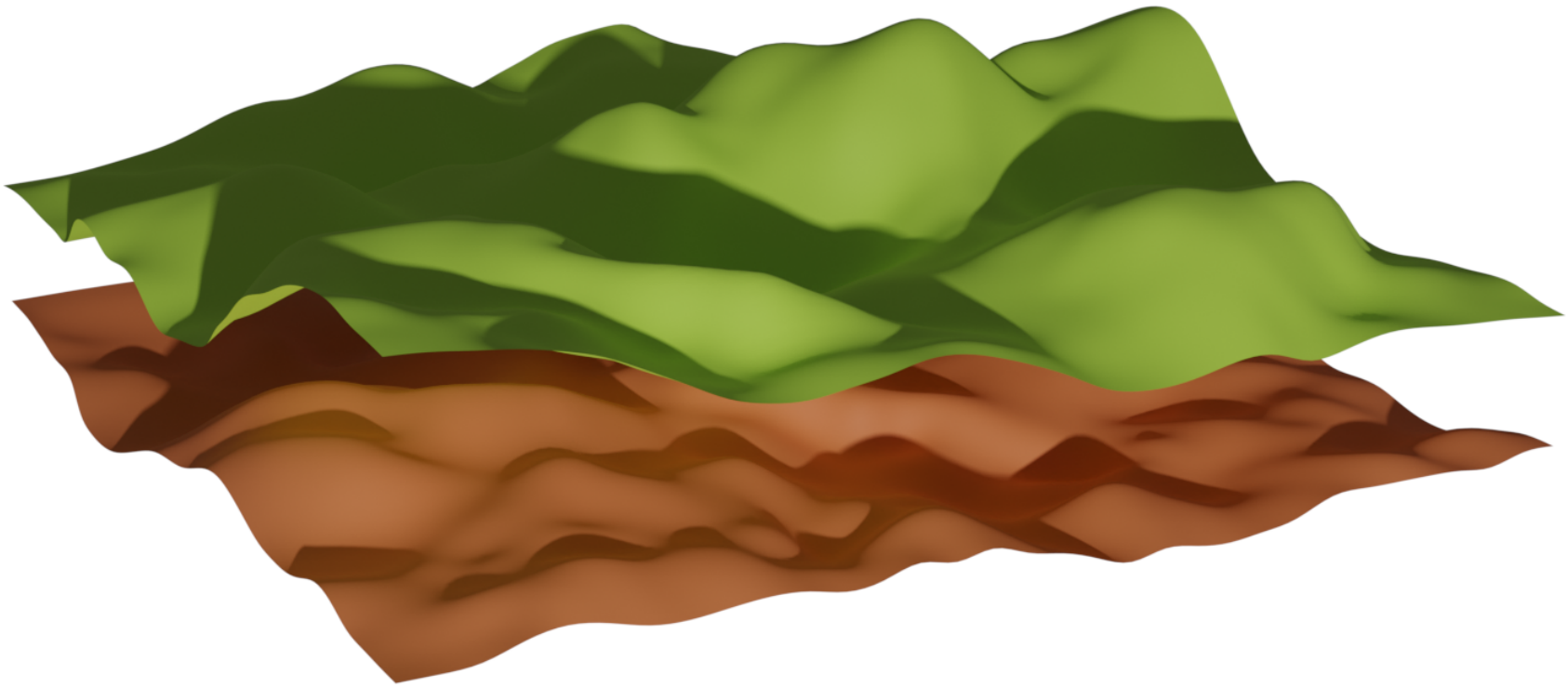
After



Positions and colors

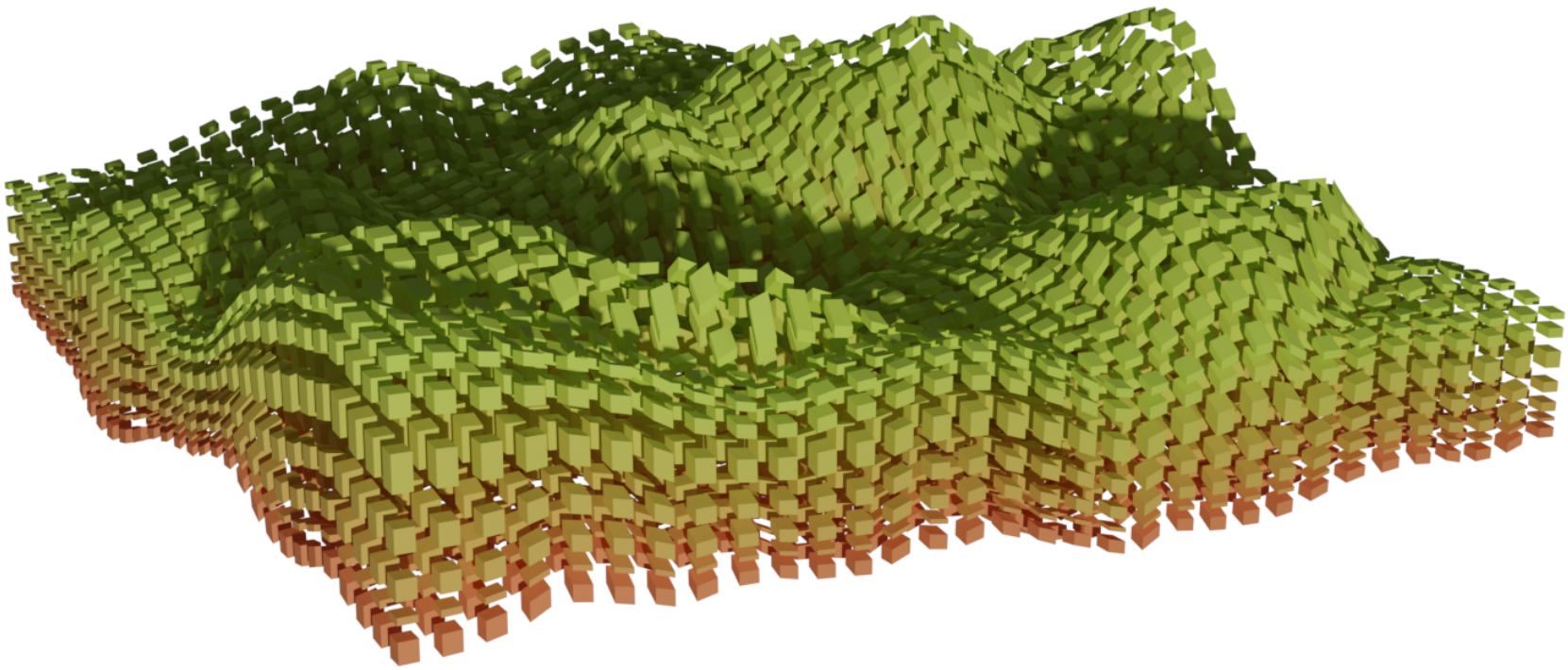
Geology inspired

Before



Geology inspired

After



Evaluation: Benchmark on Subdivision Schemes

Same input (cube) ensures each sketch has 8 unknowns, 48 eqs.
Java 11, OR-Tools 9.6, Intel Ultra 7 @ 4.8GHz, 32GB RAM.

Operation	# Expr	# Synth (%)	# Correct (%)	SolT (ms)	SynT (ms)
Surface Subdivisions					
Catmull-Clark	3	3 (100%)	1 (33%)	1.2	10
Doo-Sabin	1	1 (100%)	0 (0%)	0.4	11
Powell-Sabin	2	2 (100%)	2 (100%)	0.9	11
Blender Subdivide	2	2 (100%)	2 (100%)	0.9	9
Sierpinski Carpet	2	2 (100%)	2 (100%)	1.0	13
$\sqrt{3}$	2	2 (100%)	1 (50%)	0.9	11
Volume Subdivisions					
Menger	3	3 (100%)	3 (100%)	1.4	26
(2,2,2)-Menger	9	9 (100%)	9 (100%)	2.9	64
Surface to Volume					
Mesh to Tet	1	1 (100%)	1 (100%)	0.5	12
Mesh to Hex	3	3 (100%)	3 (100%)	1.2	16

24/28 expressions synthesized (failures outside of search space)
Full pipeline: 0.5–2 s

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.
- Arnould, Agnès et al. (2022). "Preserving consistency in geometric modeling with graph transformations". In: Mathematical Structures in Computer Science. DOI: 10.1017/S0960129522000226.
- 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 II

- 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.
- 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.

References III

- 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.
- 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.
- Jha, Susmit et al. (May 2010). "Oracle-guided component-based program synthesis". In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, pp. 215–224. DOI: 10.1145/1806799.1806833.

References IV

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

References V

- Pascual, Romain et al. (2022b). "Topological consistency preservation with graph transformation schemes". In: Science of Computer Programming. DOI: 10.1016/j.scico.2021.102728.
- 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 VI

- 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.
- Solar-Lezama, Armando (Oct. 2013). "Program sketching". In: International Journal on Software Tools for Technology Transfer 15.5, pp. 475–495. DOI: 10.1007/s10009-012-0249-7.

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.