Combinatorial maps: transformations and application to geometric modeling

Romain Pascual[1]

Agnès Arnould[2], Hakim Belhaouari[2],

and Pascale Le Gall[1]

[1] MICS University Paris-Saclay, [2] XLIM, University of Poitiers

Sept. 24, 2021

Geometric Modeling

Geometric Modeling, or how to create and edit *nD* virtual objets.

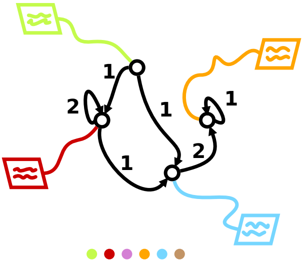
Topology-Based Geometric Modeling

Combinatorial maps represente objects through their subdivision into topological cells (volumes, faces, edges, vertices) by decreasing dimension.

The result is a graph, labeled

- on the arcs by dimensions (i.e., integers) to describe neighboring relations,
- on the nodes by embedding values (position, color, etc.) to describe the geometry.

Modeling operation defined as graph transformations.



Graph transformations : specific needs from geometric modeling

Requirements :

- Standard operations (sewing, triangulation, etc.) should be expressible as rules
- Operations should be parametrized by cells regardless of the cell size.
- Rules should preverse the model consistency
- Performance

Graph transformations : specific needs from geometric modeling

Requirements : **Solution within the Jerboa platform.**

- Standard operations (sewing, triangulation, etc.) should be expressible as rules
  - **Benchmark**
- Operations should be parametrized by cells regardless of the cell size.
  - **Rules schemes abstract cells.**
- Rules should preverse the model consistency
  - **Syntactic verification of set-theoric constraints.**
- Performance
  - **Compilation of rule schemes with optimized data structures.**

Several models

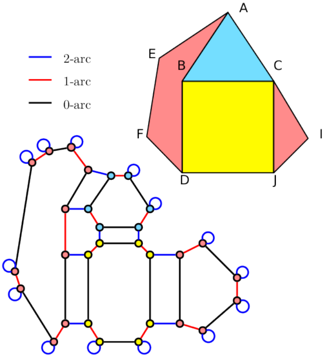
Previous works (and Jerboa) exploits generalized maps, a homoneneously defined model, easier to reason about and manipulate.

There are other models. For instance, oriented maps are more popular (supported by the CGoGN library) because of the lighter memory footprint.

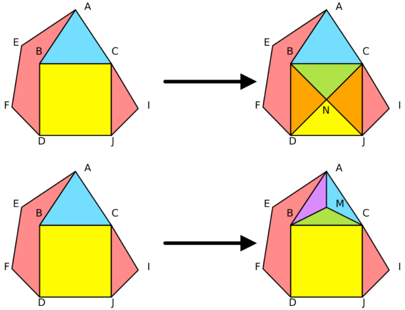
▶ Extension of previous work on topological operations to englobe both generalized and oriented maps.

Generalizing topological operations : plan

- 1/ (Embedded) Combinatorial maps as data structure.



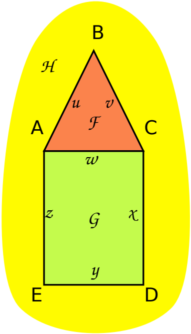
- 2/ Geometric modeling operations via rewriting.



Combinatorial maps

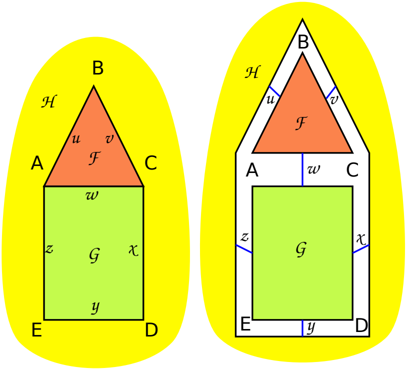
- A graph-based approach to define topological models.

Decomposition into cells (G-maps and O-maps)



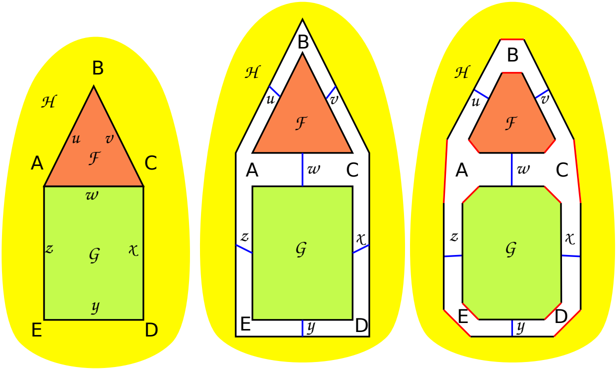
Obtained by recursive subdivision into topological cells of decreasing dimension.

Decomposition into cells (G-maps and O-maps)



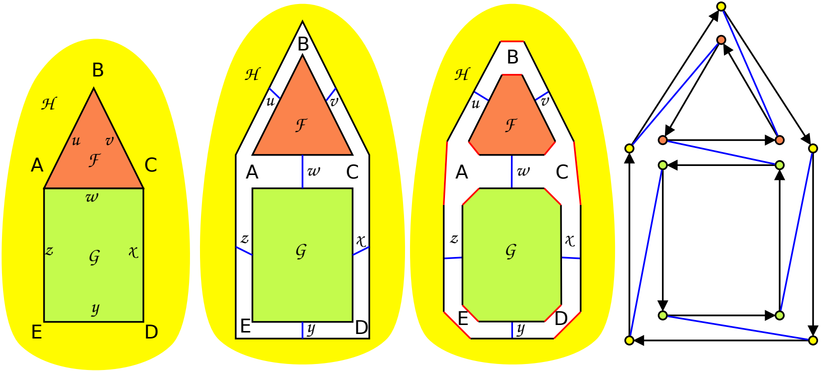
Obtained by recursive subdivision into topological cells of decreasing dimension.

Decomposition into cells (G-maps and O-maps)



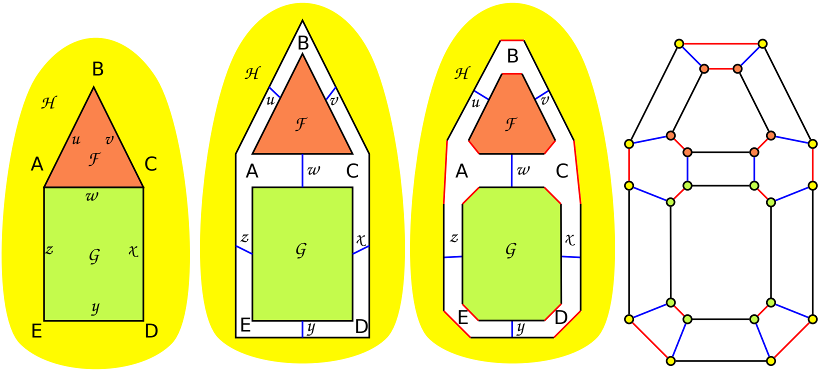
Obtained by recursive subdivision into topological cells of decreasing dimension.

Decomposition into cells (G-maps and O-maps)



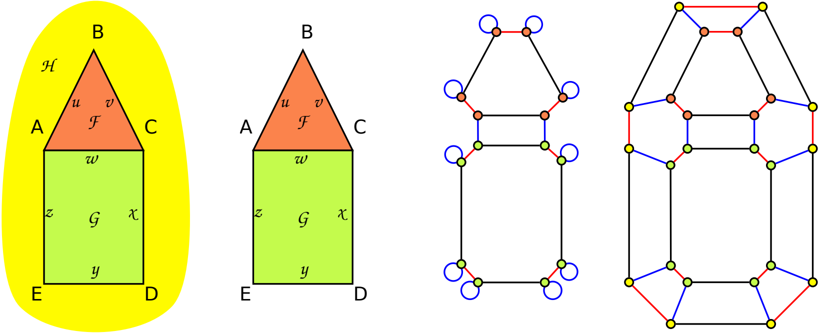
Obtained by recursive subdivision into topological cells of decreasing dimension.
Exploiting the orientation of the edges to orient the $1D$ arcs yields an oriented maps.

Decomposition into cells (G-maps and O-maps)



Obtained by recursive subdivision into topological cells of decreasing dimension.
Splitting the vertices into two nodes yields a generalized map.

Decomposition into cells (G-maps and O-maps)



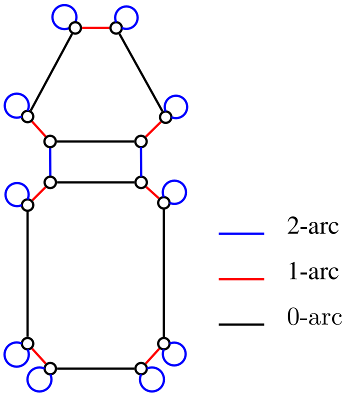
Obtained by recursive subdivision into topological cells of decreasing dimension.

Splitting the vertices into two nodes yields a generalized map.

Combinatorial maps

G-map of dimension $n \geq 0$

A graph $G = (V, E, s, t, l)$ labeled on arcs by $l : E \to [\![0, n]\!]$ such that :

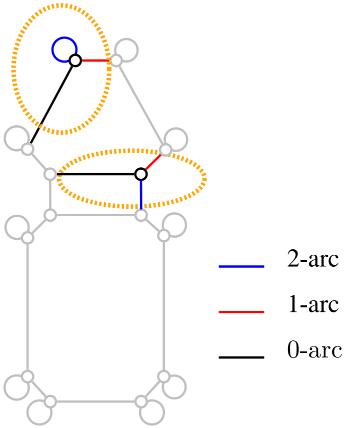


Combinatorial maps

G-map of dimension $n \geq 0$

A graph $G = (V, E, s, t, l)$ labeled on arcs by $l : E \to [\![0, n]\!]$ such that :

- **incident arcs :** each node is the source (resp. target) of a unique i-arc for $i \geq 0$.

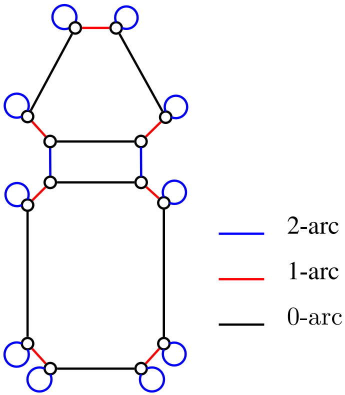


Combinatorial maps

G-map of dimension $n \geq 0$

A graph $G = (V, E, s, t, l)$ labeled on arcs by $l : E \to [\![0, n]\!]$ such that :

- **incident arcs :** each node is the source (resp. target) of a unique i-arc for $i \geq 0$.
- **non orientation :** each i-arc admits a reverse i-arc for $i \geq 0$ (G is undirected).

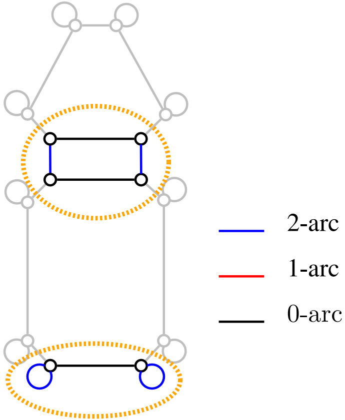


Combinatorial maps

G-map of dimension $n \geq 0$

A graph $G = (V, E, s, t, l)$ labeled on arcs by $l : E \rightarrow [\![0, n]\!]$ such that :

- **incident arcs :** each node is the source (resp. target) of a unique i-arc for $i \geq 0$.
- **non orientation :** each i-arc admits a reverse i-arc for $i \geq 0$ (G is undirected).
- **cycles :** each $ijij$-path is a cycle (for dimensions $i + 2 \leq j$).

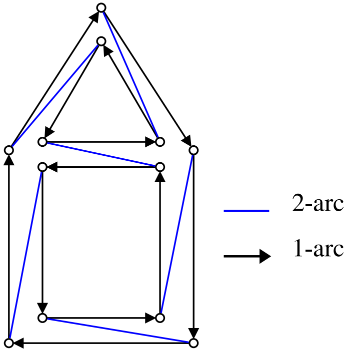


Combinatorial maps

O-map of dimension $n \geq 1$

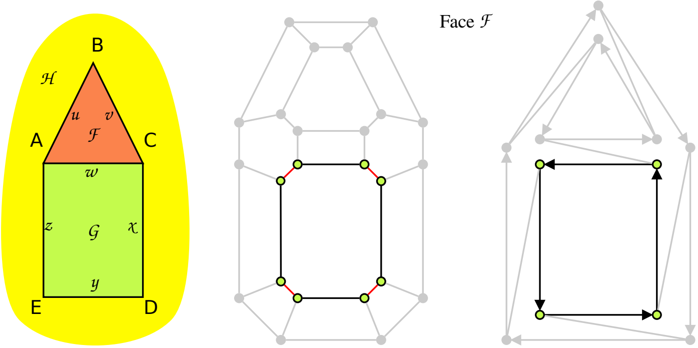
A graph $G = (V, E, s, t, l)$ labeled on arcs by $l : E \to [\![1, n]\!]$ such that :

- **incident arcs :** each node is the source (resp. target) of a unique i-arc for $i \geq 1$.
- **non orientation :** each i-arc admits a reverse i-arc for $i \geq 2$ (G is directed).
- **cycles :** each $ijij$-path is a cycle (for dimensions $i + 2 \leq j$).



Notation : $\mathbb{D}$ ($= [\![0, n]\!]$ or $[\![1, n]\!]$) is the set of dimensions (i.e., the labeling alphabet).

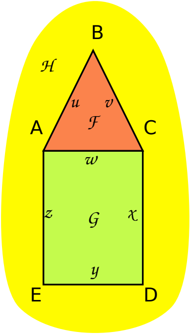
Cells and Orbits

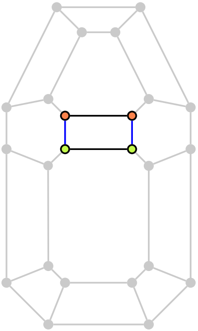


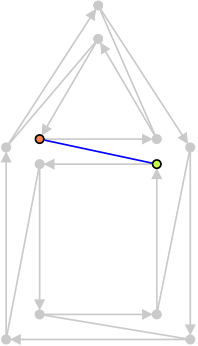
We can retrieve the object's cells using words of dimensions.

- G-map : $(0 + 1)^*$.
- O-map : 1^*.

Cells and Orbits



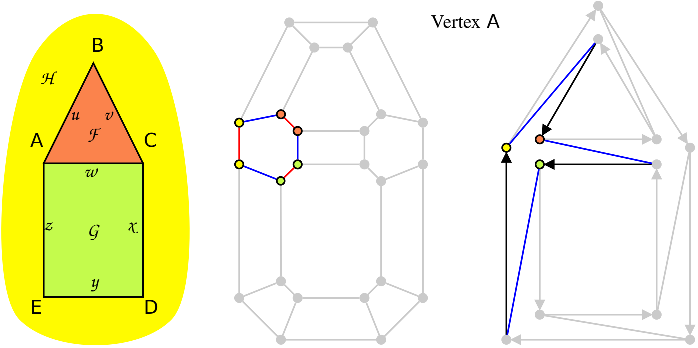




We can retrieve the object's cells using words of dimensions.

- G-map : $(0+2)^*$.
- O-map : 2^*.

Cells and Orbits



We can retrieve the object's cells using words of dimensions.

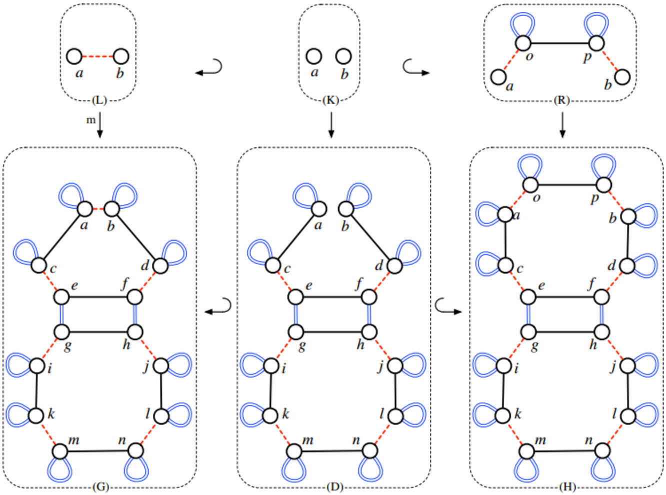
- G-map : $(1+2)^*$.
- O-map : $(21)^*$.

Notation : $\mathbb{W}$ will denote a finite langage on $\mathcal{D}^*$ (i.e., a finite set of words on the labeling alphabet).

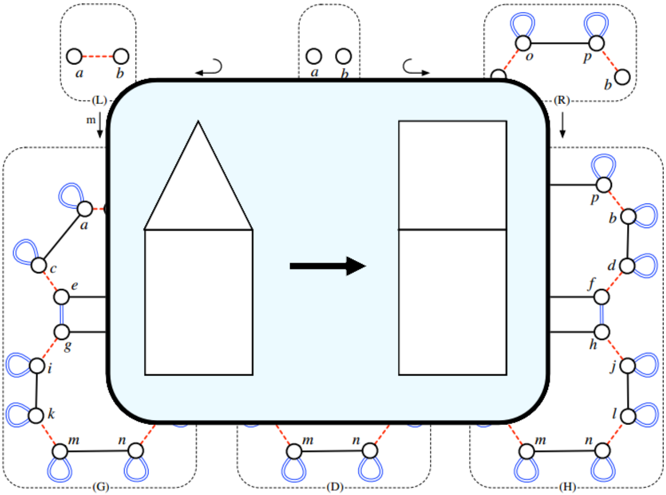
Modeling operations

- Graph rewriting and issues for application in geometric modeling.

Topological transformation as DPO rewriting

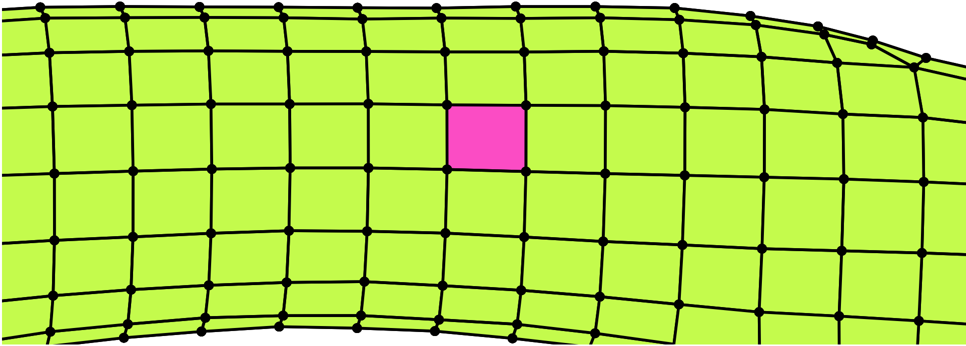


Topological transformation as DPO rewriting



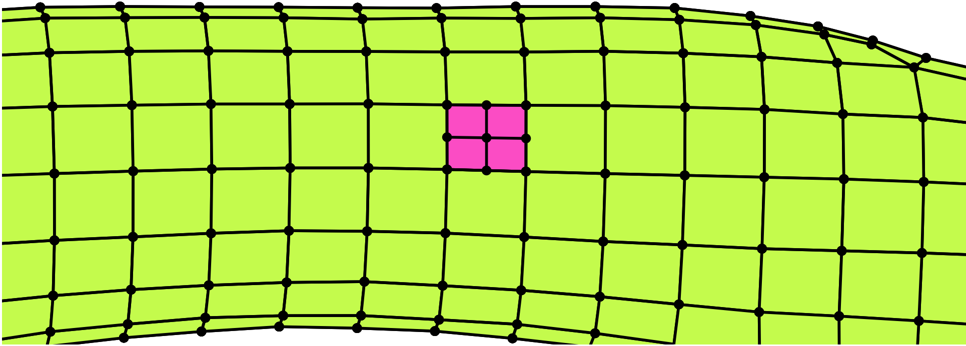
Genericity

- A formalization of modeling operations



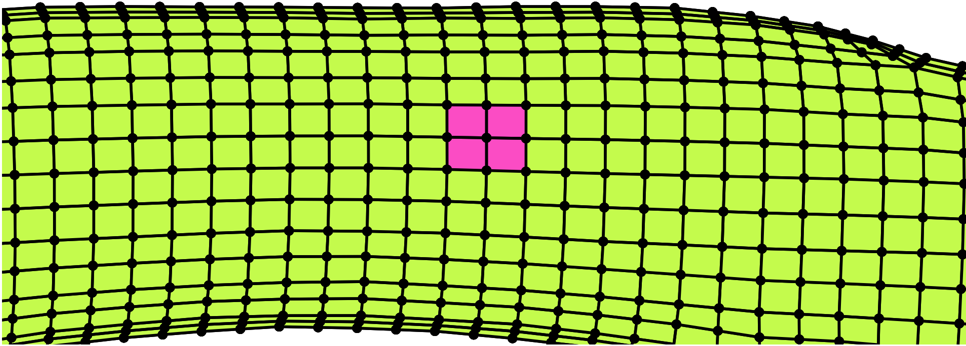
Genericity

- A formalization of modeling operations

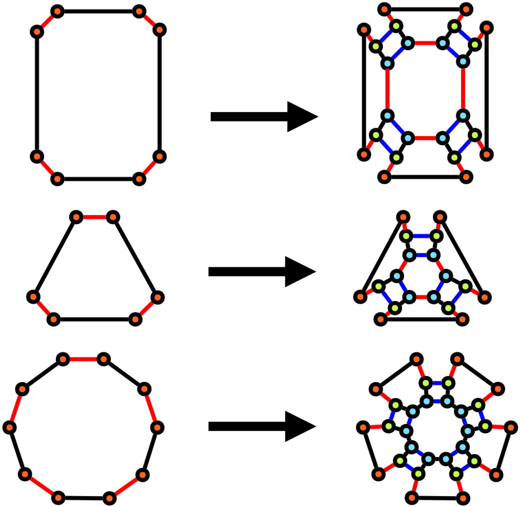


Genericity

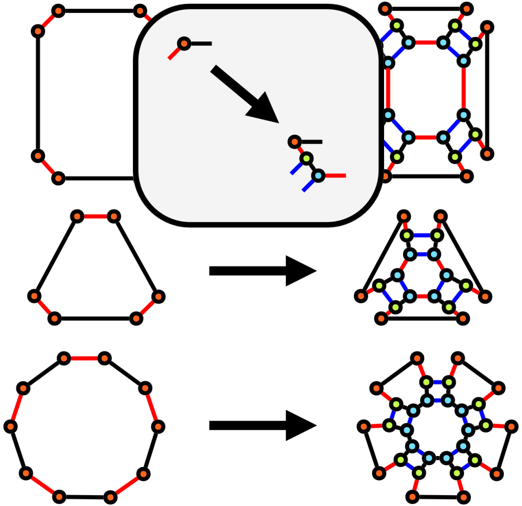
- A formalization of modeling operations



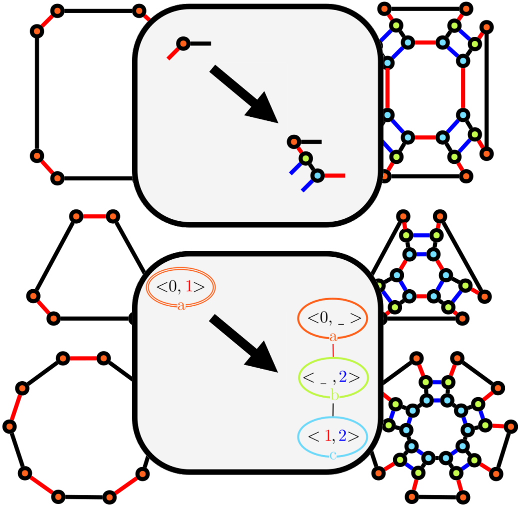
Orbit rewriting (variables)



Genericity



Genericity



Some remarks

Orbit variables and orbit type rewriting extends rule to subparts of the modeled object.

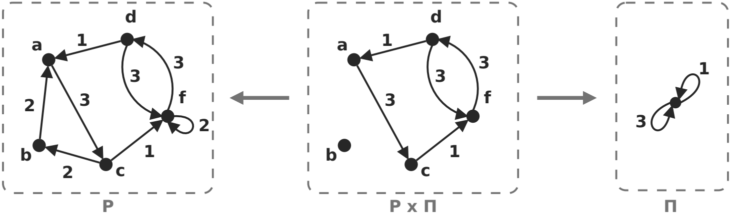
- Issue : not suited to deal with oriented maps (orbits described by words)

The approach with orbit variables implicitely exploits a product construction that we can formalize and generalize to pathes.

Global operations using product : arc deletion

Pullback on the terminal element (i.e., product) allows to model global modification of graphs :

- Arc deletion based on labels : deletion of 2 while preserving the other labels.

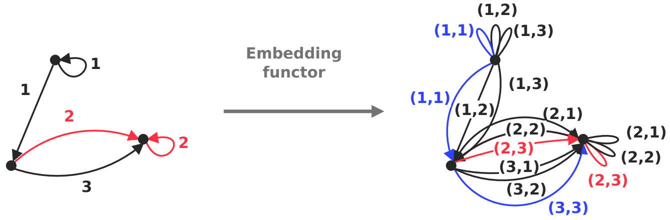


Global operations using product : relabeling

Pullback on the terminal element (i.e., product) allows to model global modification of graphs :

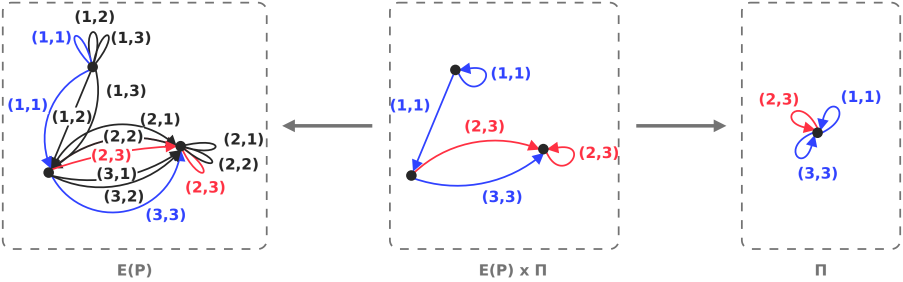
- Arc relabeling based on labels : relabeling $2 \mapsto 3$ while preserving the other labels can be described by the relation $\{(1,1),\ (2,3),\ (3,3)\}$.

Embedding Functor



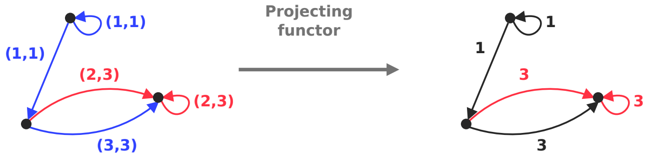
The embedding functor $\mathbb{E}_\Sigma : \Sigma\text{-Graph} \to (\Sigma^2)\text{-Graph}$ transforms an i-arc ($i \in \Sigma$) into $|\Sigma|$ arcs labeled (i, j) for each $j \in \Sigma$, making each relabeling possible.

Product simulates fonction application



Construction of the product (pullback on the terminal element).

Projecting Functor



The projecting functor $\pi_\Sigma : (\Sigma^2)\text{-Graph} \to \Sigma\text{-Graph}$ keeps the second part of arc labels (i.e., the relabeled part).

Summary

The construction is summarized by the following commutative diagram, for a labeling alphabet Σ :

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

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

Summary

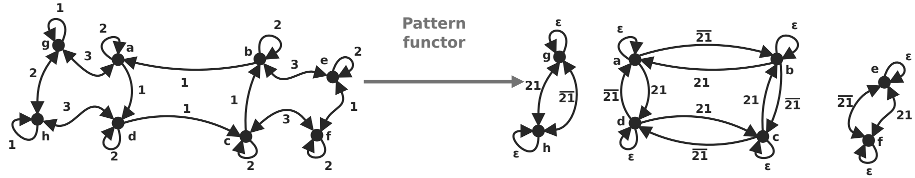
The construction is summarized by the following commutative diagram, for a labeling alphabet Σ :

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

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

▶ Replace Π with "any" $(\mathbb{W} \times \mathbb{D})$-graph.

Pattern Functor



The pattern functor $\mathbb{P} : \mathbb{D}\text{-Graph} \to \mathbb{W}\text{-Graph}$ maps a graph G to the graph $\mathbb{P}(G)$ that has the same nodes as G and, for $w \in W$ (path labels), a w-arc of source v_1 and target v_2 whenever there is a w-path $v_1 \overset{w}{\rightsquigarrow} v_2$ in G.

$\mathbb{D}$ is extended to incorporate $\overline{d}$ for a d dimension d to represent reverse traversal of arcs.
$\mathbb{W}$ is extended accordingly.

Application of a rule scheme

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

- The input consists of a G-map (or O-map) $\mathcal{G}$ and a rule scheme $L^\bullet \hookleftarrow K \hookrightarrow R$.

Application of a rule scheme

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

- The pattern functor builds pathes for the given langage.

Application of a rule scheme

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

- The monomorphism p_v extract the connected component P_v in $\mathbb{P}(G)$ that contains a given node v.

Application of a rule scheme

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

- The embedding functor adds every possible relabeling.

Application of a rule scheme

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

- Pullback/Product constructions

Application of a rule scheme

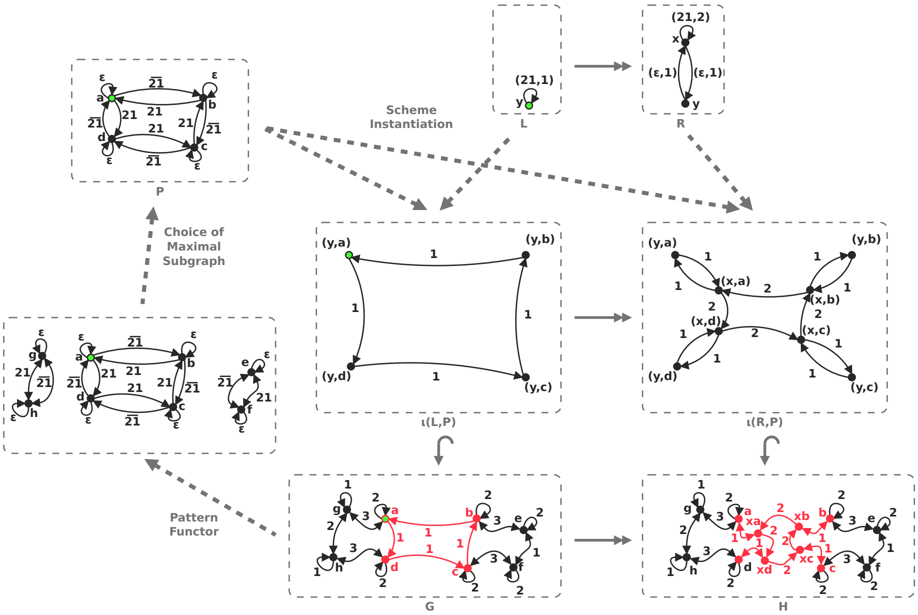
$$
\begin{tikzcd}
1_{\mathbb{W}\times\mathbb{D}} & & & \\
E_{\mathbb{D}}(P_v) \arrow[u, "!_{E_{\mathbb{D}}(P_v)}"] & L^\bullet \arrow[ul, "!_{L^\bullet}"'] & K \arrow[l, hook'] \arrow[r, hook] \arrow[ull, "!_K"'] & R \arrow[ulll, "!_R"'] \\
P_v \arrow[u, Rightarrow, "E_{\mathbb{D}}"] \arrow[d, hook,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

- The projecting functor keeps only the relabeling of the arc.

Application of a rule scheme

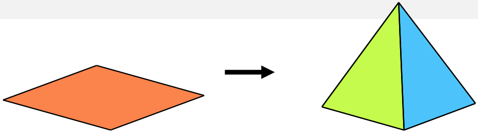
$$
\begin{tikzcd}
1_{\mathbb{W}\times\mathbb{D}} & & & \\
E_{\mathbb{D}}(P_v) \arrow[u, "!_{E_{\mathbb{D}}(P_v)}"] & L^\bullet \arrow[ul, "!_{L^\bullet}"'] & K \arrow[l, hook'] \arrow[r, hook] \arrow[ull, "!_K"'] & R \arrow[ulll, "!_R"'] \\
P_v \arrow[u, Rightarrow, "E_{\mathbb{D}}"] \arrow[d, hook,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

- Standard DPO rewriting.

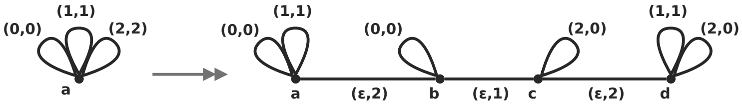

(21,1)
y
L
(21,2)
x
(ε,1)
(ε,1)
y
R
P
Scheme Instantiation
Choice of Maximal Subgraph
Pattern Functor
(y,a)
(y,b)
(y,d)
(y,c)
ι(L,P)
(x,a)
(x,b)
(x,d)
(x,c)
ι(R,P)
G
H

Examples

Cone operation.

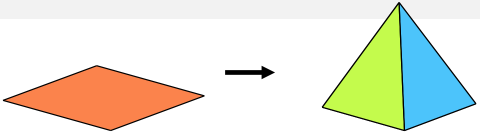


- G-map rule scheme

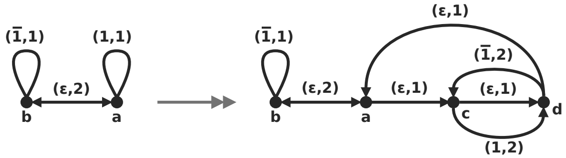


Examples

Cone operation.



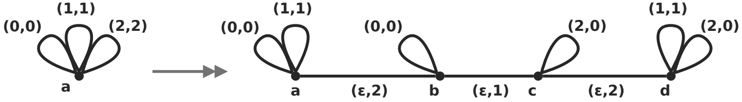
- O-map rule scheme



Examples

Edge rounding operation.

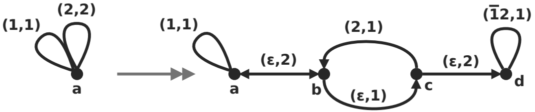
- G-map rule scheme



Examples

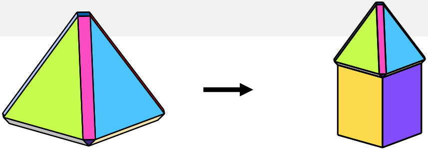
Edge rounding operation.

- O-map rule scheme

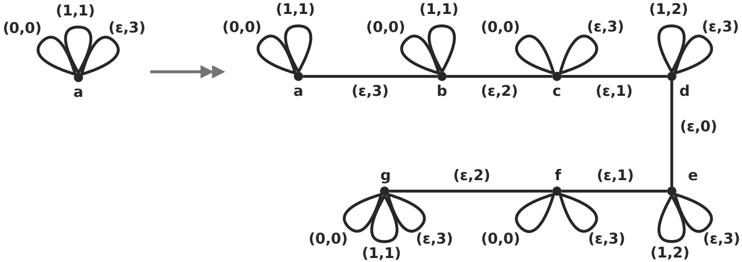


Examples

Face extrusion operation.

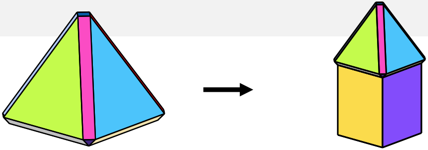


- G-map rule scheme

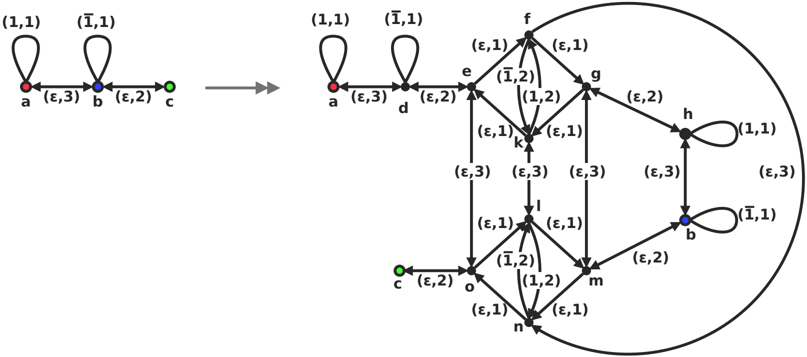


Examples

Face extrusion operation.



- O-map rule scheme



Consistency preservation

- Constraints on the topological relations : soundness of the structure, e.g., angles are correctly formed, vertices are incident to edges.
- Constraints on the embedding values : soundness of the geometry, e.g. all nodes that belong to the same vertex have the same position.

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

▶ Graph transformations are enriched with conditions to preserve these consistency properties.

Requirement : Provide feedback to the rule designer.

Topological consistency

Topological constraints (incident arcs, non-orientation and cycles) : first-order logic.

Rule schemes are compiled into optimized code. To be efficient, **no computation about the consistency preservation can be done when the scheme rule is applied**.

Topological consistency

Topological constraints (incident arcs, non-orientation and cycles) : first-order logic.

Rule schemes are compiled into optimized code. To be efficient, **no computation about the consistency preservation can be done when the scheme rule is applied**.

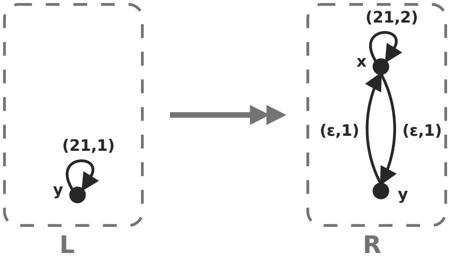
A scheme rule is considered valid whenever all its possible instantiations preserve the model consistency.

- Set-theoric conditions on rule schemes.

Gluing condition

In 'standard' DPO-rule application, the gluing condition (existence of a pushout complement) ensures the applicability of a rule.

Monic matches reduce the gluing condition to the dangling condition : no node of $m(L) \setminus m(K)$ is source or target of an arc in $G \setminus m(L)$.

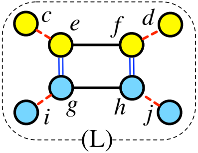


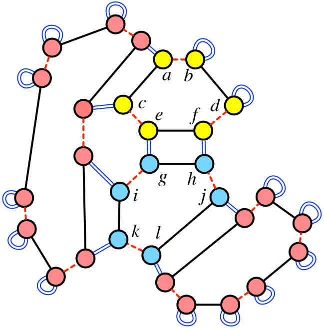
When G satisfies the incident arcs constraint on $\mathbb{D}$, the gluing condition is equivalent to **all deleted nodes of $L \setminus K$ are deleted with one arc per dimension**.

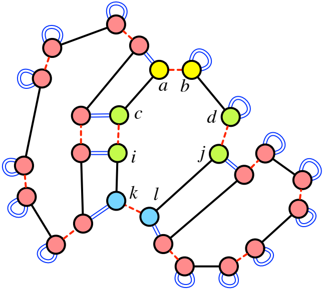
▶ This condition **extends to scheme rule** when considering the first part of the label.

Breaking the geometric consistency

Constraint : all nodes of an orbit supporting an embedding should have the same value.

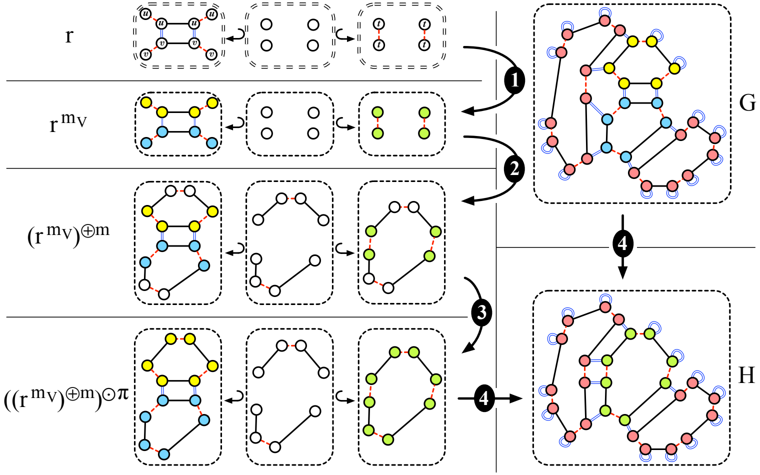






Restoring the geometric consistency

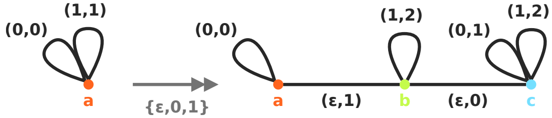
- Topological and geometric extensions (extended to rules with variables using equivalence on terms.)



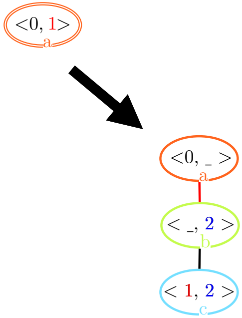
Simulating orbit rewriting

Product-based graph transformations subsumes orbit variables.

- Rule scheme (G-map)



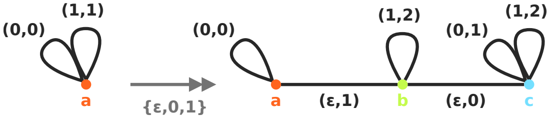
- Rule with variables



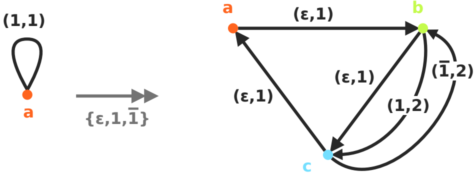
Simulating orbit rewriting

Product-based graph transformations subsumes orbit variables.

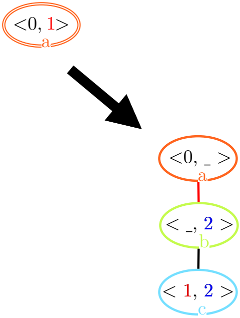
- Rule scheme (G-map)



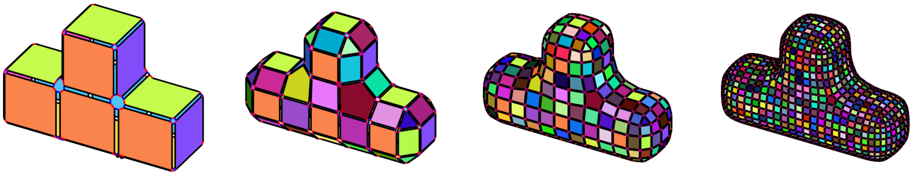
- Rule scheme (O-map)

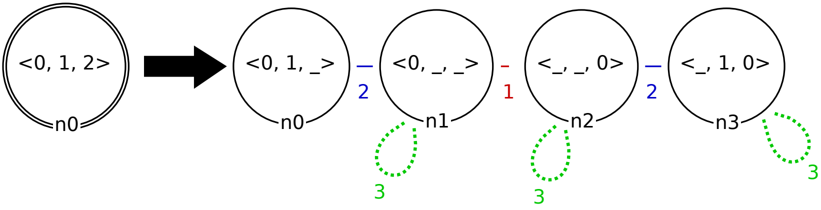


- Rule with variables

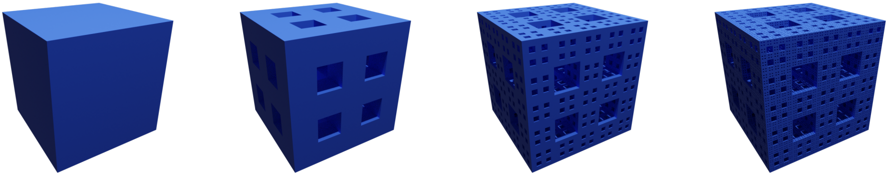


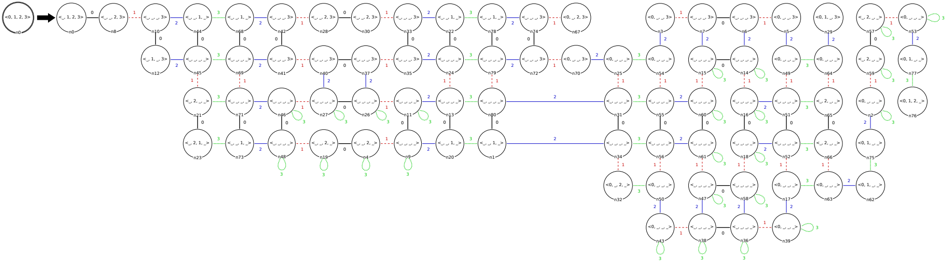
Doo-Sabin [Doo and Sabin, 1978]





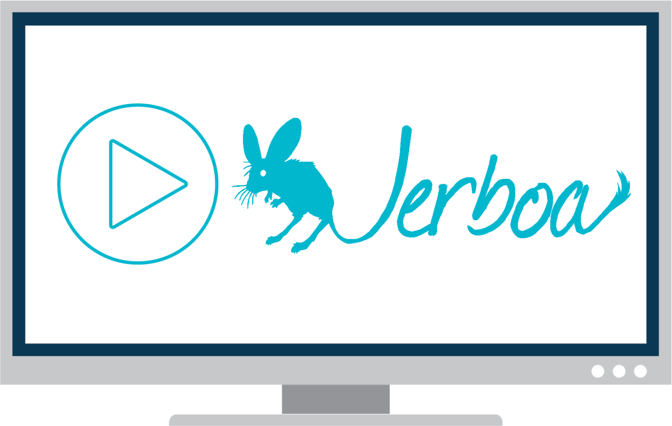
Menger sponge $(2, 2, 2)$ [Richaume et al., 2019]





Jerboa

http ://xlim-sic.labo.univ-poitiers.fr/jerboa/



Doo, D. and Sabin, M. (1978).
Behaviour of recursive division surfaces near extraordinary points.
Computer-Aided Design, 10(6) :356–360.

Richaume, L., Largeteau-Skapin, G., Zrour, R., and Andres, E. (2019).
Unfolding Level 1 Menger Polycubes of Arbitrary Size With Help of Outer Faces.
In Discrete Geometry for Computer Imagery (DGCI), Paris, France.

Jerboa's architecture