A Category of Plane Graphs — Extended Abstract

Malin Altenmüller   Ross Duncan

We introduce categorical structure for plane graphs, which represent non-symmetric monoidal categories where swaps of wires are not permitted. We define notions of boundary graph and partitioning span, which capture exactly the situations in which we want to apply rewrite rules. We show that the necessary properties for double pushout rewriting are met by these structures. From the notion of graph we briefly summarize how we construct the category of rotation systems on top of the category of graphs, enabling topology preserving rewriting of graphs.

Introduction

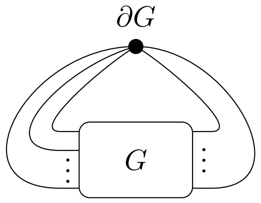
String diagrams [14] are a graphical formalism to reason about monoidal categories. Equational reasoning in symmetric string diagrams can be implemented as graph or hyper-graph rewriting subject to various side conditions to capture the precise flavour of the monoidal category intended [4, 5, 6, 11, 2, 1]. In this work we address the *non-symmetric* setting. Abstractly, this more general setting can capture both the symmetric case and the braided monoidal one, admitting the possibility to reason about a larger class of theories. A more practical motivation comes from the area of quantum computing, where string diagrams are often used to model quantum circuits [3], their connectivity restrictions imposed by the qubit architecture require a theory without implicit SWAP gates.

We introduce a category of planar graphs and explain its properties suitable for double pushout rewriting [8]. This is an important step towards formalising non-symmetric string diagrams and their rewriting theory.

Plane Graphs

By plane graph we are referring to an *embedding* of a graph into the plane (or, equivalently, onto a sphere), also known as map.

Diagrams with input and outputs can be characterised as open graphs, which themselves we represent by graphs with *boundary vertices*: we connect all inputs and outputs to an additional vertex (the graph's boundary vertex ∂G), preserving their order.



Combinatorically, embeddings of graphs into surfaces can be represented by *rotation systems*, which store for each vertex its order of edges incident. Rotation systems determine the embedding of a graph on a surface [10, 7, 9]. (Rotation systems in general can accommodate graphs embedded in any orientable surface, not merely the plane. Though at this moment we are only addressing the plane case.)

A Category for Plane Graphs

We define a category of directed graphs which will serve as the underlying category for rotation systems. It is not plane by default, but designed to accommodate and enforce topology restrictions by adding rotation systems on top of graphs at a later point.

A total graph is a functor $G : (\bullet \rightrightarrows \bullet) \to \mathbf{Set}$. Concretely, a graph is a pair of functions assigning source and target vertices to edges.

$$\begin{tikzcd} E \arrow[r, shift left, "s"] \arrow[r, shift right, "t"'] & V \end{tikzcd}$$

A morphism of graphs is a pair of maps f_V, f_E such that the following squares commute.

$$\begin{tikzcd} E \arrow[r, "f_E"] \arrow[d, "s"'] & E' \arrow[d, "s'"] \\ V \arrow[r, "f_V"'] & V' \end{tikzcd} \qquad \begin{tikzcd} E \arrow[r, "f_E"] \arrow[d, "t"'] & E' \arrow[d, "t'"] \\ V \arrow[r, "f_V"'] & V' \end{tikzcd}$$

To be able to replace the boundary vertex with a context, the notion of morphism has to admit partial vertex maps. This comes with two challenges, the first one being to actually represent these partial maps. We are moving towards the category of partial graphs and maps $[\bullet \rightrightarrows \bullet, \mathbf{Pfn}]$, though this is not quite enough. Commutation of the corresponding squares in this category is strict, meaning it includes equality of the domains of definition. We address this issue by using the poset enrichment of $\mathbf{Pfn}$,and work in the category $[\bullet \rightrightarrows \bullet, \mathbf{Pfn}]_{\leq}$ of functors and *lax* natural transformations:

$$\begin{tikzcd} E \arrow[r, "f_E"] \arrow[d, "s"'] \arrow[dr, phantom, "\leq"] & E' \arrow[d, "s'"] \\ V \arrow[r, "f_V"'] & V' \end{tikzcd} \qquad \begin{tikzcd} E \arrow[r, "f_E"] \arrow[d, "t"'] \arrow[dr, phantom, "\leq"] & E' \arrow[d, "t'"] \\ V \arrow[r, "f_V"'] & V' \end{tikzcd} \tag{1}$$

While the lax category $[\bullet \rightrightarrows \bullet, \mathbf{Pfn}]_{\leq}$ is better than the previous attempt, it does not meet all of our requirements. Because vertices in these graphs represent morphisms in a monoidal category, their arity has to be preserved by any morphism. As we allow for vertices *without* any edges attached, a condition on edges alone does not suffice. We introduce an additional requirement, on vertices:

$$\begin{tikzcd} V \arrow[r, "f_V"] \arrow[d, "s^{-1}"'] \arrow[dr, phantom, "\geq"] & V' \arrow[d, "s'^{-1}"] \\ P(E) \arrow[r, "P(f_E)"'] & P(E') \end{tikzcd} \qquad \begin{tikzcd} V \arrow[r, "f_V"] \arrow[d, "t^{-1}"'] \arrow[dr, phantom, "\geq"] & V' \arrow[d, "t'^{-1}"] \\ P(E) \arrow[r, "P(f_E)"'] & P(E') \end{tikzcd} \tag{2}$$

where s^{-1} and t^{-1} are the preimage maps of s and t respectively, and P is the powerset functor.

The second issue introduced by using placeholder vertices is concerning edges: If the context graph includes a self loop (e.g. a cup or cap), substituting a graph into this context will connect two of its edges with each other. This is where injectivity of the edge component of a morphism has to fail. Contrarily, some form of injectivity on edges needs to be present, as we are interpreting all morphisms as embeddings, and especially for preserving the arity of a vertex. Instead of injectivity on edges we use injectivity on the connection points, or *flags*, of an edge with its source and target vertices.

Definition. A *flag* is a pair of an edge and a vertex (e, v) where $v = s(e)$ or $v = t(e)$. The *flag map* induced by the graph map is the pair (f_E, f_V).

The flag map is a partial map, it is undefined on a (e, v), whenever f_V is undefined on v. We call the property of the flag map being injective *flag injectivity.*

Theorem. Flag injectivity, together with Equation 2 implies the preservation of vertex degree.

Proof Sketch. Preservation of vertex degree means $|s^{-1}(v)|+|t^{-1}(v)| = |s^{-1}(f_V(v))|+|t^{-1}(f_V(v))|$. If f_V is defined on a vertex v, the flag map is total on all flags incident to v. It also implies the *strict* commutation of the diagrams in Equation 2. Because the flag map is injective, the number of edges attached to v is preserved. □

We now have introduced all the necessary structure to define the category of graphs:

Definition. Let $\mathcal{G}$ be the category whose objects are total graphs $G = (V, E, s, t)$ and whose morphisms $f : G \to G'$ consist of a partial injective function $f_V : V \to V'$ and a total function $f_E : E \to E'$, such that Equation 2 holds for morphisms and such that the flag map induced by f_E and f_V is injective.

The additional requirements introduced with Equation 2 and the property of flag injectivity, are all describing a subcategory of the lax functor category we started with:

Proposition. $\mathcal{G}$ is a subcategory of $[\bullet \rightrightarrows \bullet, \mathbf{Pfn}]_{\leq}$.

Categorical Properties for Rewriting

Double pushout rewriting [8] is an approach to formalising the equations of an equational theory on monoidal categories by rewrite rules

$$\begin{tikzcd}
L \arrow[d, "m"'] \arrow[dr, phantom, "\urcorner", very near start] & B \arrow[l, "l"'] \arrow[r, "r"] \arrow[d] & R \arrow[d] \arrow[dl, phantom, "\ulcorner", very near start] \\
G & C \arrow[l] \arrow[r] & H
\end{tikzcd}$$

The rewrite rule asks for graph L to be replaced by R, while B ensures that both graphs have the same connectivity. $m : L \to G$ is the *embedding* of L into G, and H is the resulting graph from executing the rewrite $L \to R$ in G. C is the *context graph*: G with L removed. In the algebraic graph literature the notion of adhesive category [12, 13] is commonly used, it is a class of categories in which DPO rewriting behaves well.
The category of directed graphs $\mathcal{G}$ is very specific, and in general not adhesive. Therefore we are going to restrict the type of rewrite rule to the specific case we are interested in and show, that it meets the necessary adhesive properties, namely the existence of pushouts and the existence and uniqueness of pushout complements.

$$\begin{tikzcd} L \arrow[d, "m"'] & B \arrow[l, "l"'] \arrow[d, "c"] \\ G & C \arrow[l, "g"] \arrow[ul, phantom, "\ulcorner", very near end] \end{tikzcd}$$

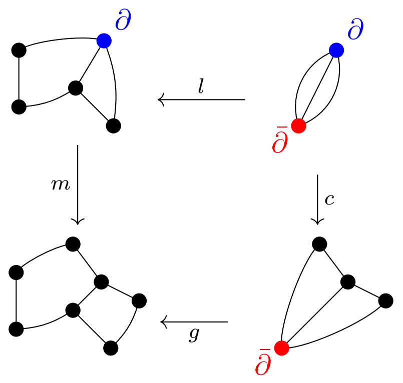


Figure 1: Example of a pushout of a partitioning span. (Directions of edges are omitted for simplicity.)

Boundary Graphs and Partitioning

The boundary vertex of each graph stores its interface, its outer connectivity. When substituting the graph into a bigger one, the boundary vertex gets replaced by the new environment. The boundary vertex can be interpreted as the placeholder for a context graph.

We are continuing this interpretation when we describe the graphs in a DPO rewriting diagram. Besides the boundary vertex which acts as a placeholder for the *context*, we introduce the *dual* boundary vertex, being the placeholder for the graph itself. Using vertices as placeholders does not only simplify our notion of graphs — all graphs are *total* —, but mainly allows for imposing rotations on the vertices later on, preserving the topological properties of a graph. With the notions of boundary graph and its dual, we now define the components for rewriting, representing exactly the cases which we are interested in.

Definition. A *boundary graph* is a pair of vertices, the *boundary vertex* and its *dual*, connected by a set of edges. A *partitioning span* is a pair of maps $L \xleftarrow{l} B \xrightarrow{c} C$ in $\mathcal{G}$, where B is a boundary graph, the vertex component l_V is undefined on the boundary vertex, and c_V is undefined on the dual boundary vertex.

An example of a partitioning span and its pushout in $\mathcal{G}$ is depicted in Figure 1. The name *partitioning span* arises from the fact that each of the maps out of the boundary graph replaces one half of it. Hence each graph has an *outside* and an *inside*, connected via the edges present in the boundary graph.

Theorem. In $\mathcal{G}$, pushouts of partitioning spans exist.

Proof Sketch. The candidate for the pushout graph G is:

- $V_G = (V_C + V_L) \setminus V_B$,
- $E_G = (E_L + E_C)/\!\sim$ where $\sim$ is the least equivalence relation such that $l_E(e) = c_E(e)$ for $e \in E_B$,
- $s_G(e) = \begin{cases} s_C(e), & \text{if } e \in E_B, l_V(s_B(e)) \text{ defined} \\ s_L(e), & \text{if } e \in E_B, c_V(s_B(e)) \text{ defined} \\ s_C(e), & \text{if } e \notin E_B, e \in E_C \\ s_L(e), & \text{if } e \notin E_B, e \in E_L \end{cases}$, similarly for t_G,

- m_V undefined on any $v \in V_B$,
- g_V undefined on any $v \in V_B$.

□

For calculating the pushout complement, we introduce the corresponding structure to partitioning spans. In this pair of edges, first the dual boundary vertex and then the boundary vertex get replaced, meaning that we are defining a graph and then embedding it into a bigger one.

Definition. A *boundary preserving embedding* is a pair of maps $B \xrightarrow{l} L \xrightarrow{m} G$ in $\mathcal{G}$, where B is a boundary graph, l_V is undefined on the boundary vertex and m_V is undefined on the dual boundary vertex.

Theorem. In $\mathcal{G}$, pushout complements of boundary preserving embeddings exist and are unique.

Proof Sketch. The candidate for the pushout complement C is:

- $V_C = (V_G + V_B) \setminus V_L$,
- $E_C = E_G \setminus (E_L \setminus E_B)$,
- $s_C = \begin{cases} s_B(e), & \text{if } e \in E_B, l_V(s_B(e)) \text{ undefined} \\ s_G(e), & \text{otherwise} \end{cases}$, similarly for t_C,
- c_V is undefined on $v \in V_B$ whenever $l_V(v)$ is defined,
- g_V is undefined on any $v \in V_B$.

□

Towards a Category of Rotation Systems

The construction of the category of rotation systems is going to take two steps: firstly we are going to equip the graphs in the above defined category with rotation information. This introduces an ordering on the edges incident to a vertex, Equation 2 now holds on *cyclic lists of edges* rather than sets. Secondly we apply the action of a forgetful functor, recovering rotation systems for undirected graphs. The translation of the categorical properties required for rewriting through both these steps works due to the way we set up the underlying category. Thus rewriting for rotation systems makes sense.

References

[1] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Pawel Sobocinski & Fabio Zanasi (2016): *Rewriting modulo symmetric monoidal structure.* In: *LiCS'16 Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pp. 707–719, doi:10.1145/2933575.2935316.

[2] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Pawel Sobocinski & Fabio Zanasi (2017): *Confluence of Graph Rewriting with Interfaces.* In: *ESOP'17.*

[3] Bob Coecke, Ross Duncan, Aleks Kissinger & Quanlong Wang (2015): *Generalised Compositional Theories and Diagrammatic Reasoning.* In G. Chirabella & R. Spekkens, editors: *Quantum Theory: Informational Foundations and Foils*, Springer, doi:10.1007/978-94-017-7303-4_10.

[4] Lucas Dixon & Ross Duncan (2009): *Graphical Reasoning in Compact Closed Categories for Quantum Computation. Annals of Mathematics and Artificial Intelligence* 56(1), pp. 23–42, doi:10.1007/s10472-009-9141-x.

[5] Lucas Dixon, Ross Duncan & Aleks Kissinger (2010): *Open Graphs and Computational Reasoning.* In: *Proceedings DCM 2010*, *Electronic Proceedings in Theoretical Computer Science* 26, pp. 169–180, doi:10.4204/EPTCS.26.16.

[6] Lucas Dixon & Aleks Kissinger (2013): *Open Graphs and Monoidal Theories. Math. Structures in Comp. Sci.* 23(2), pp. 308–359, doi:10.1017/S0960129512000138.

[7] J. R. Edmonds (1960): *A combinatorial representation for polyhedral surfaces. Notices of the American Mathematical Society* 7(A646).

[8] Hartmut Ehrig, Karsten Ehrig, Ulrike Prange & Gabriele Taentzer (2006): *Fundamentals of Algebraic Graph Transformation.* Monographs in Theoretical Computer Science, Springer Berlin Heidelberg, doi:10.1007/3-540-31188-2.

[9] Jonathan L. Gross & Thomas W. Tucker (2001): *Topological Graph Theory.* Dover.

[10] Lothar Heffter (1891): *Über das Problem der Nachbargebiete. Mathematische Annalen* 38(4), pp. 477–508, doi:10.1007/BF01203357.

[11] Aleks Kissinger & Vladimir Zamdzhiev (2015): *Equational Reasoning with Context-Free Families of String Diagrams.* In Francesco Parisi-Presicce & Bernhard Westfechtel, editors: *Graph Transformation*, *Lecture Notes in Computer Science* 9151, Springer International Publishing, pp. 138–154, doi:10.1007/978-3-319-21145-9_9. Available at `http://dx.doi.org/10.1007/978-3-319-21145-9_9`.

[12] S. Lack & P. Sobocinski (2003): *Adhesive categories.* Technical Report BRICS RS-03-31, BRICS, Department of Computer Science, University of Aarhus.

[13] Stephen Lack & Pawel Sobocinski (2005): *Adhesive and quasiadhesive categories. Theoretical Informatics and Applications* 39(3), pp. 511 – 545, doi:10.1051/ita:2005028.

[14] Peter Selinger (2011): *A survey of graphical languages for monoidal categories.* In Bob Coecke, editor: *New structures for physics*, *Lecture Notes in Physics* 813, Springer, pp. 289–355, doi:10.1007/978-3-642-12821-9_4.