

A CATEGORY OF SURFACE-EMBEDDED GRAPHS

Malin Altenmüller

malin.altenmuller@strath.ac.uk

GR_ETA Seminar, 8th September 2023

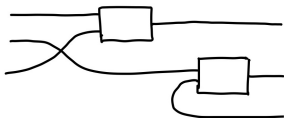
About me

- PhD Student at the University of Strathclyde in Glasgow, in the Mathematically Structured Programming Group (supervised by Ross Duncan & Conor McBride)
- Research Intern at Huawei R&D in Edinburgh, working with graphs as intermediate compiler representation (working with Dan Ghica)

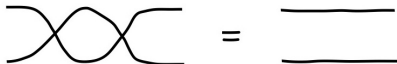
Theory and Implementation of Topology-Aware String Diagrams:

- reasoning with diagrams — categories of graphs
- data types for diagrams (and their reasoning) in Agda

String Diagrams



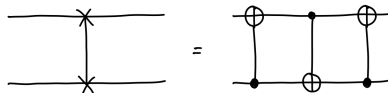
- syntax for monoidal categories
- composition both ways easy for the diagram
- represent computational processes
- observe specific properties of the monoidal category, e.g. symmetric monoidal category (SMC)



Topology-Aware String Diagrams

Interested in *non-symmetric* theories:

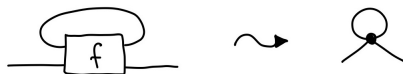
- string diagrams for quantum, swaps are non-trivial:



- other example: printing circuits
- general theory to instantiate to symmetry, braiding, etc.

Graphs

- reasoning on the diagrams themselves (rewrite rules)
- need a combinatorial representation of them
- using graphs,
translating generators into vertices, wires into edges:

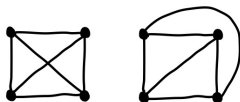


- reasoning by graph rewriting, categorically

Graphs of non-symmetric diagrams?

Graph Embeddings

- drawing of a graph onto a surface



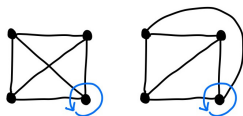
- planar graph if it has a plane embedding



Rotation Systems

How to define a graph embedding combinatorially?

- rotation system: store the order of incident edges at each vertex



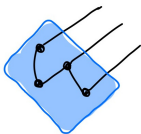
Theorem

A rotation system uniquely determine a graph embedding.

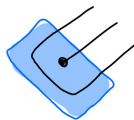
- plan: define a suitable category of graphs, then add rotation information

An Example of DPO Rewriting

given: rewrite rule

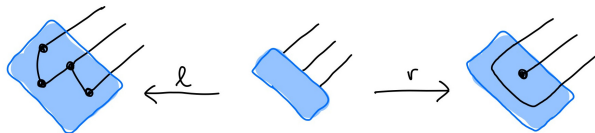


\Rightarrow



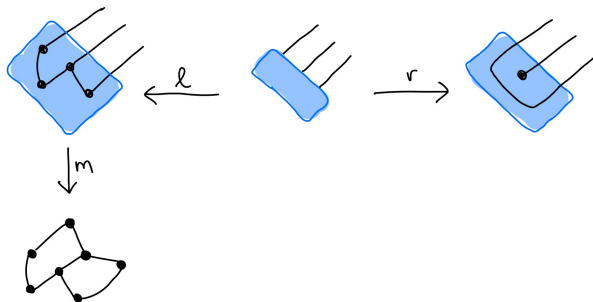
An Example of DPO Rewriting

given: rewrite rule as span with common boundary in the middle



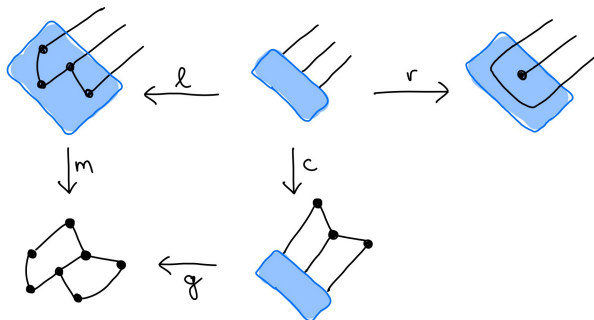
An Example of DPO Rewriting

given: matching of the LHS within a graph



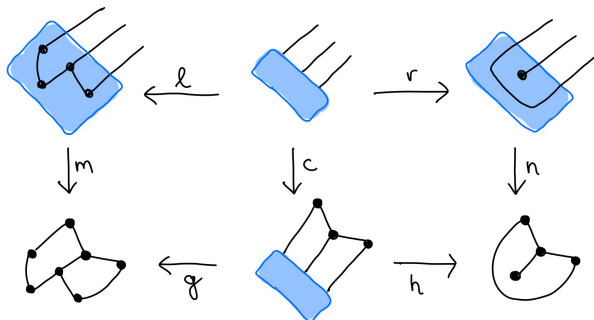
An Example of DPO Rewriting

construct: context graph by pushout complement



An Example of DPO Rewriting

construct: final graph by pushout



DPO Rewriting

$$\begin{array}{ccccc} L & \longleftarrow & B & \longrightarrow & R \\ \downarrow & \lrcorner & \downarrow & \lrcorner & \downarrow \\ G & \longleftarrow & G \setminus L & \longrightarrow & G[R/L] \end{array}$$

- rewrite rules are $L \Rightarrow R$, with common boundary B
- double-pushout diagram, all maps are embeddings
- required: pushouts, pushout complements, notion of embedding (related: adhesive categories¹)

¹Lack and Sobociński, "Adhesive categories".

DPO Rewriting

$$\begin{array}{ccccc} L & \longleftarrow & B & \longrightarrow & R \\ \downarrow & \lrcorner & \downarrow & & \downarrow \\ G & \longleftarrow & G \setminus L & \longrightarrow & G[R/L] \end{array}$$

Observation:

- $C = G \setminus L$: context with a hole

DPO Rewriting

$$\begin{array}{ccccc} G \setminus C & \longleftarrow & B & \longrightarrow & R \\ \downarrow & \lrcorner & \downarrow & & \downarrow \\ G & \longleftarrow & C & \longrightarrow & G[R/L] \end{array}$$

Observation:

- $C = G \setminus L$: context with a hole
- $L = G \setminus C$: LHS with a “hole”

Standard Category of Graphs

We start from the standard category of graphs:

- graphs are $E \begin{array}{c} \xrightarrow{s} \\ \xrightarrow{t} \end{array} V$
- morphisms $(E \begin{array}{c} \xrightarrow{s} \\ \xrightarrow{t} \end{array} V) \rightarrow (E' \begin{array}{c} \xrightarrow{s'} \\ \xrightarrow{t'} \end{array} V')$ are pairs of an edge map $f_E : E \rightarrow E'$ and a vertex map $f_V : V \rightarrow V'$, such that:

$$\begin{array}{ccc} E & \xrightarrow{f_E} & E' \\ s \downarrow & & \downarrow s' \\ V & \xrightarrow{f_V} & V' \end{array} \qquad \begin{array}{ccc} E & \xrightarrow{f_E} & E' \\ t \downarrow & & \downarrow t' \\ V & \xrightarrow{f_V} & V' \end{array}$$

Remark

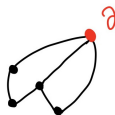
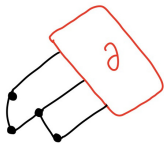
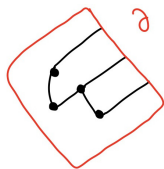
(Almost) all graphs are drawn undirected in this presentation.

Open Graphs

- process diagrams have inputs and outputs
- but also potentially holes
- encode dangling edges in the theory
- different approaches:
open graphs, representative vertices, cospans²
- morphisms for open graphs don't preserve the surface
- cannot add rotation information to loose edges

²Baez and Courser, *Structured Cospans*.

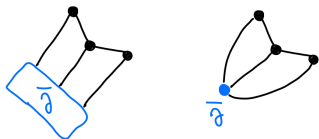
Boundary Vertex



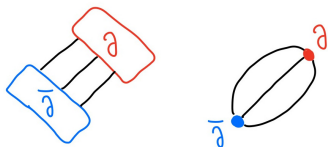
- identify the “outside” of a graph
- attach input and output edges to this region
- outside is one region of the graph
- replace the region with a *boundary vertex*
- all graphs are total
- I can add rotation information to each vertex later

Dual Boundary Vertex

- other regions of the graphs may have dangling edges

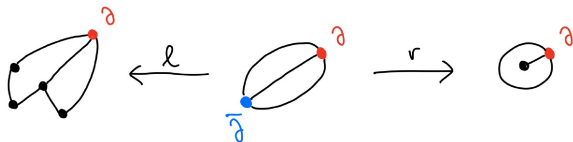


- replace *each* of them by a vertex
- graph with an outside *and* one hole

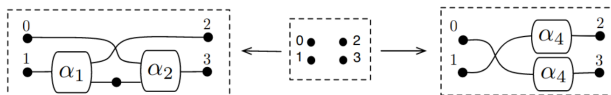


Rewrite Rules

a rewrite rule now looks like this:



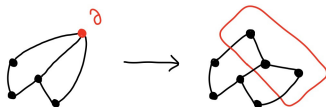
related: double pushout rewriting of graphs with interfaces³



³Bonchi et al., "Confluence of Graph Rewriting with Interfaces".

Requirements for Graph Morphisms

- vertex map needs to be partial

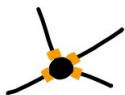


- edge map is cannot be injective

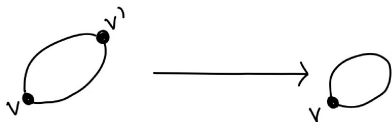


What is the right notion of embedding then?

Flags and Flag Maps



- connection points between vertices and their incident edges, pairs (v, e)
- flag map (f_E, f_V) *partial* map induced by graph map
- characterise morphisms/embeddings on the flag map



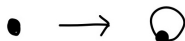
- example: flag injectivity

Flag Surjectivity

Starting with the condition for standard graph morphisms $(V, E) \rightarrow (V', E')$:

$$\begin{array}{ccc} E & \xrightarrow{f_E} & E' \\ s \downarrow & & \downarrow s' \\ V & \xrightarrow{f_V} & V' \end{array}$$

What about vertices with no edges attached?

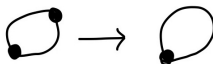


Flag Surjectivity

Condition on *vertices*, by considering the preimage:

$$\begin{array}{ccc} V & \xrightarrow{f_V} & V' \\ s^{-1} \downarrow & & \downarrow s'^{-1} \\ P(E) & \xrightarrow{P(f_E)} & P(E') \end{array}$$

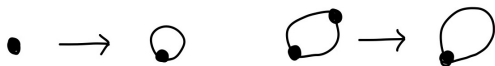
What about vertices where f_V is undefined?



Flag Surjectivity

Flag surjectivity = lax commutation of the square:

$$\begin{array}{ccc} V & \xrightarrow{f_V} & V' \\ s^{-1} \downarrow & \geq & \downarrow s'^{-1} \\ P(E) & \xrightarrow{P(f_E)} & P(E') \end{array}$$



Graphs with Circles

Objects are total graphs, as defined above

Morphisms are (f_E, f_V) where

- f_E is total
- the flag map is surjective
(no increase of flags at a vertex)

+ other conditions

Graph *embeddings* are

- flag injective (no decrease of flags at a vertex)

+ other conditions

It's a category!

Remark on Circles

- might have edges without source or target: loops



- but flag surjectivity is defined on *vertices*
- edge set is $E + O$, with O being the circles
- maybe we need more structure for loops?

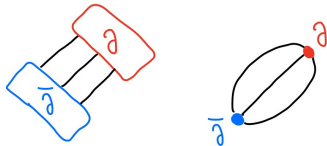
Rewriting for Graphs with Circles

The category of graphs is not adhesive!

But it has enough adhesive properties for the case we're interested in:

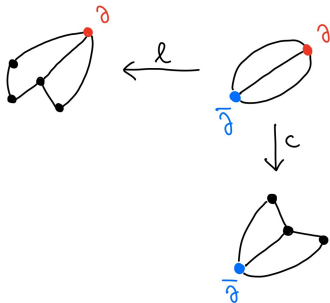
Boundary Graph

= boundary vertex and dual boundary vertex, connected by edges



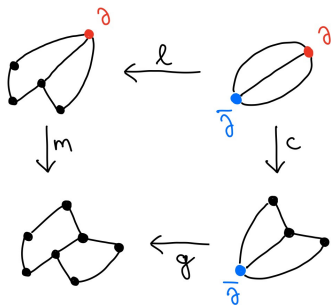
Partitioning Spans

partition a graph into two (connected) parts: context and subgraph



Partitioning Spans

partition a graph into two (connected) parts: context and subgraph

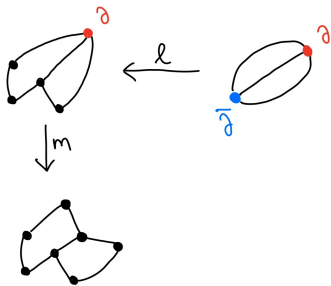


Theorem

Pushouts of partitioning spans exist, and all morphisms in the pushout square are embeddings.

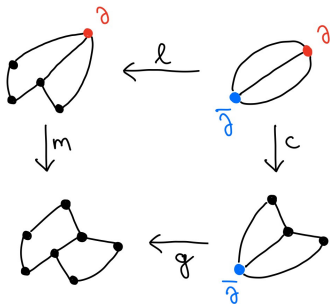
Boundary Embeddings

for constructing pushout complements which give rise to partitioning spans



Boundary Embeddings

for constructing pushout complements which give rise to partitioning spans

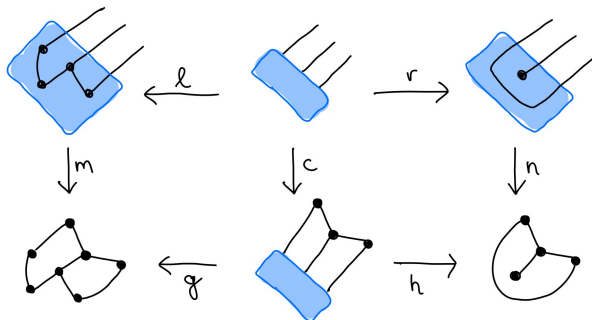


Theorem

Pushout complements of boundary embeddings exist and are unique (up to degeneracies).

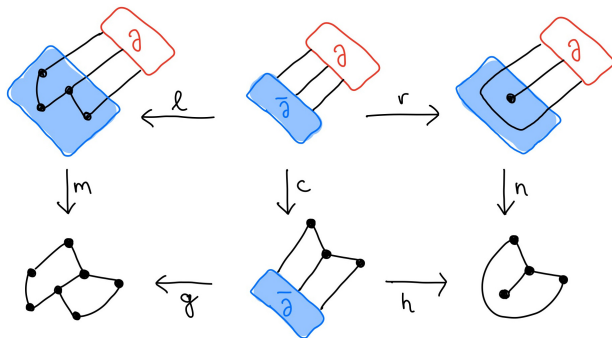
The Same Example of DPO Rewriting

Remember this example?



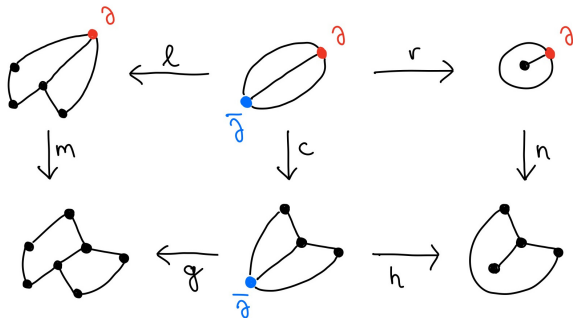
The Same Example of DPO Rewriting

Let's add some boundary regions ...



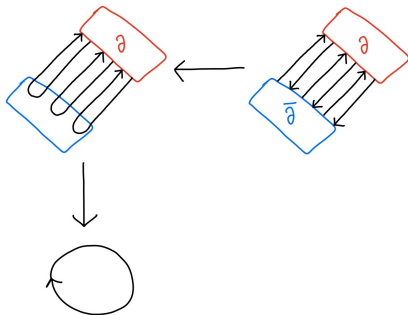
The Same Example of DPO Rewriting

... and use their representative vertices



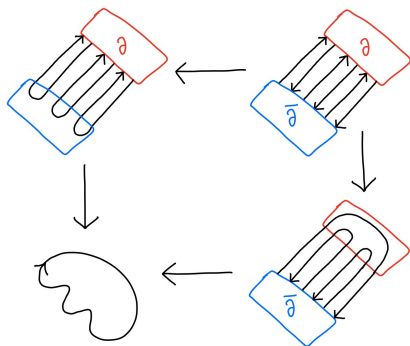
Let's talk about Loops!

problem: construct a pushout complement of a loop



Let's talk about Loops!

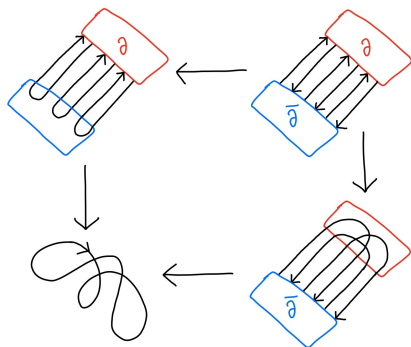
problem: construct a pushout complement of a loop



has a plane solution

Let's talk about Loops!

problem: construct a pushout complement of a loop



has a plane solution
and a non-plane solution

Category of Rotation Systems

obj: graphs + cyclic ordering of flags for all vertices

arr: same as graphs + order preservation condition

Example

$$\begin{array}{ccc} V & \xrightarrow{f_V} & V' \\ t^{-1} \downarrow & \geq & \downarrow t'^{-1} \\ P(E) & \xrightarrow{P(f_E)} & P(E') \end{array}$$

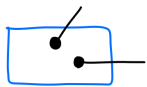
$$\begin{array}{ccc} V & \xrightarrow{f_V} & V' \\ t^{-1} \downarrow & \geq & \downarrow t'^{-1} \\ CList(E) & \xrightarrow{CList(f_E)} & CList(E') \end{array}$$

Theorem

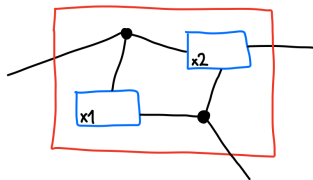
Pushouts and pushout complements are the same as in the underlying category of graphs.

The Operad of Plane Graphs

An instance of David Spivak's Operad of Wiring Diagrams⁴



$$g : 0 \mapsto 2$$

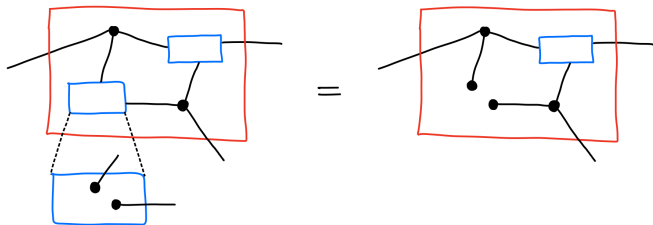


$$f : (0 \mapsto 2, 2 \mapsto 1) \rightarrow (1 \mapsto 2)$$

⁴Spivak, *The operad of wiring diagrams: formalizing a graphical language for databases, recursion, and plug-and-play circuits.*

The Operad of Plane Graphs

Composition corresponds to substitution
(aka rewriting in the category of rotation systems):

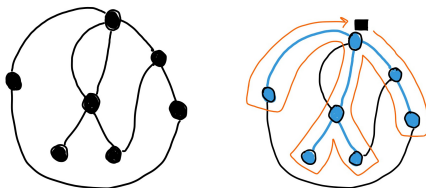


$$f \circ_1 g : (2 \mapsto 1) \rightarrow (1 \mapsto 2)$$

Implementation

Developing a data type of these kind of graphs in the dependently-typed programming language Agda⁵.

- define graphs inductively along their spanning trees

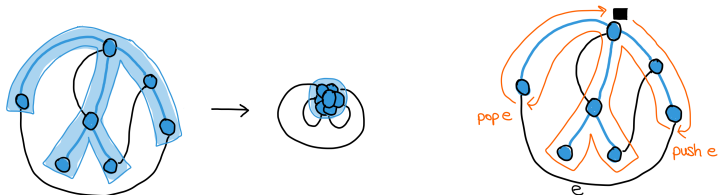


- store edges not in the spanning tree in a structure alongside it
- interested in notions of focussing and context graph

⁵MA and McBride, "A Datatype of Planar Graphs (Abstract)".

Implementation: Planar Graphs

- structure of the additional edges determine the surface
- can contract the spanning tree and observe



- in the plane case, a stack is exactly what we need!

What about higher genus graph embeddings?

Summary






- non-symmetric string diagrams are interesting
- fix inputs and outputs to control topology – boundary vertices
- restrict your rewrite rules to meaningful cases
- category of graphs with circles extendable to rotation systems

Ideas

- What about surface-embedded loops?
- What about more complex boundary graphs?
- What about implementing higher genus graphs?

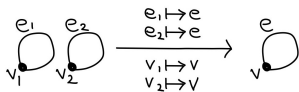
THANK YOU FOR YOUR ATTENTION!

Bibliography

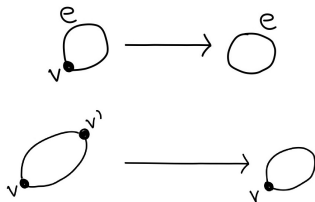
-  Baez, John C. and Kenny Courser. *Structured Cospans*. 2020. arXiv: 1911.04630 [math.CT].
-  Bonchi, Filippo et al. “Confluence of Graph Rewriting with Interfaces”. In: *Programming Languages and Systems*. Ed. by Hongseok Yang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 141–169.
-  Lack, Stephen and Paweł Sobociński. “Adhesive categories”. In: *International Conference on Foundations of Software Science and Computation Structures*. Springer. 2004, pp. 273–288.
-  MA and Conor McBride. “A Datatype of Planar Graphs (Abstract)”. In: *28th International Conference on Types for Proofs and Programs*. 2022.
-  Spivak, David I. *The operad of wiring diagrams: formalizing a graphical language for databases, recursion, and plug-and-play circuits*. 2013. arXiv: 1305.0297 [cs.DB].

Appendix: Examples

Valid morphisms:

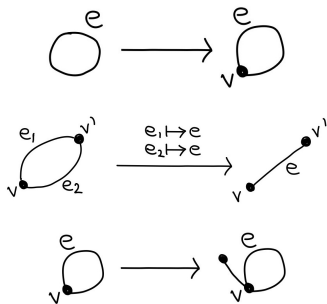


Embeddings:



Appendix: Non-Examples

These aren't morphisms in the category:



Appendix: Definition Graphs with Circles

A morphism $f : G \rightarrow G'$ between two graphs with circles consists of two (partial) functions $f_V : V \rightarrow V'$ as above, and $f_A : A \rightarrow A'$, satisfying the conditions listed below. Note that any such f_A factors as four maps,

$$\begin{array}{ll} f_E : E \rightarrow E' & f_{EO} : E \rightarrow O' \\ f_{OE} : O \rightarrow E' & f_O : O \rightarrow O' \end{array}$$

The following conditions must be satisfied:

- $f_A : A \rightarrow A'$ is total;
- the component $f_{OE} : O \rightarrow E'$ is the empty function;
- the pair (f_V, f_E) forms a flag surjection between the underlying graphs.

If, additionally, the following three conditions are satisfied, we call the morphism an *embedding*:

- $f_V : V \rightarrow V'$ is injective;
- the component f_O is injective;
- the pair (f_V, f_E) forms a flag bijection between the underlying graphs.