From Geometry to Interaction

Valentin Maestracci, Thomas Seiller

Submitted to

From Geometry to Interaction

Valentin Maestracci [a,1] Thomas Seiller[b,2]

[a] *IRISA/CNRS – Spicy Team*
Rennes, France

[b] *CNRS, IRL 3527 – JFLI*
Tokyo, Japan

Abstract

Linear Realisability is a program of research in the line of Geometry of Interaction, where models of computations are used to create models of Linear Logic. This work aims to solve some of the problems of the underlying computational models, and more precisely related to their representation of program execution, by realising them as directed topological spaces. This construction formally shows the decomposition into three parts of the dynamics of computing normal forms: glueing, saturation, hiding. Gluing composes programs without loss of information, saturation computes the information represented by the space, and finally hiding removes partial computations. This shows that execution transforms coSpans/Pushouts – purely geometric information with implicit computational content –, into Spans/Pullbacks – explicit functional/relational information, or a normal form. Finally, it explains the so-called 2-cocycle (or *trefoil*) property [32] of interaction graphs as a reflection of geometric associativity. All of this opens the link with directed algebraic topology and glueing as a potential way to creates new models of Linear Realisability.

Keywords: Linear Realisability, Linear Logic, Geometry of Interaction, Execution Formula, Interaction Graphs, category, semi-category, compact closed category, traced monoidal category, Int-construction, directed algebraic topology, cobordisms, coSpan, Span, pushout, pullback, quiver, path, directed Space, glueing, saturation, hiding

1 Introduction

Geometry of Interaction (GoI) was a research programme initiated by Girard [13] to try and define models of linear logic [12] that would not be purely denotational but retain computational content. Many instance of such models were defined since the introduction of the programme, and one of its research successor can be found in Seiller's Linear Realisability [31,22], in which models of linear logic are created from models of computations, where proofs are interpreted by programs (compiled into, even), thus retaining their computational content, and where formulas/types are interpreted as sets of programs sharing similar behaviours (with respect to their interaction with other programs). The simplest model of reference for linear realisability is the model of Interaction Graphs [25], which originated as a simplification of Girard's GoI in the Hyperfinite Factor [14], and was later generalized by Seiller in many directions [28,29,30] to extend the logical expressivity that the programs generates in their behaviours.

In Geometry of Interaction and Linear Realisability, the operation of *execution*, obtained through the so-called "execution formula" is the way to compute the normal form of programs (or, equivalently, model

⋆ Thomas Seiller and Valentin Maestracci were partially funded by the ANR DySCo project ANR-22-CE48-0003.
[1] Email: valentin.maestracci.academic@proton.me
[2] Email: thomas.seiller@cnrs.fr.

program execution). In terms of categories, this corresponds to a composition of morphism. We first investigate how the simple model of interactions graphs [25] can be reformulated in the traditional setting of traced monoidal categories [19] in a category **ioGrph**. This category suffers of a traditional problem of interaction graphs, namely the loss of topological information during execution, which makes the model somehow incomplete.

Because of this loss, the construction of logic from the model of interaction graphs depends on a property called the 2-cocycle (or *trefoil*) property [28,32], which is a generalisation of the usual adjunction property of linear logic. It is indeed the condition guaranteeing the associativity of execution when the model is extended into it sister model, the model of *projects*, which is defined by adjoining programs with an element in a monoid representing this information, called a *wager*.

But why forget something to add it again later on? Trying to find a way to not lose it in the first place, one can notice that such a construction was not necessary in some other geometric context, such as the category of cobordisms. Additionally, this geometric understanding of the models' dynamics seemed to suggest that the execution should not be considered primitive, but rather the entanglement of three operations: glueing, saturation and hiding.

In this paper we develop a formal treatment of these geometric intuitions by proposing a geometric realisation of graphs as directed topological spaces. We introduce the semi-category **ioQuiv** and the category **ioSpace**, the latter based on directed algebraic topology, both having glueing on boundaries as composition. Notably, the category **ioSpace** elegantly avoids the problem of the wager, as it does not loose topological information anymore.

We then link back our newly defined categories to **ioGrph**, by formally decomposing execution in two steps. First, we apply a semi-functor $d\Pi_1$ which associates to a space its fundamental category as a morphism in a newly defined semi-category **ioCat**. This corresponds to a step of saturation: computing of all possible paths in the space. Second, we apply a functor $\mathcal{H}\mathtt{ide} : \mathbf{ioCat} \to \mathbf{ioGrph}$, corresponding to a step of suppression of all paths whose endpoints do not belong to a boundary (which correspond to partial, or intermediary, steps of computation).

As a byproduct, this shows that the execution in Interaction graphs and GoI models can be understood as a way to turn a geometric object (coSpans), whose computational content is implicit, into mathematical relations (Spans), whose content is explicit. Finally, our work also establishes that the 2-cocyle property is a reflection of associativity at a geometric level, as was foreseen by Seiller in his PhD [25].

2 Introduction to Interaction Graphs

We start by a quick overview of the basic notions involved in the interaction graphs model.

Definition 2.1 A directed graph G is a tuple (V_G, E_G, s_G, t_G), where V_G is a finite set of vertices, E_G is the set of edges, and s_G, t_G – the source and target maps – are functions from E_G to V_G. An interaction graph is simply a directed graph, and we will refer to interaction graphs simply as graphs since we will (almost) not consider undirected graphs. Two graphs can be glued into a single graph $G \amalg H$ (read G "glue" H), by taking the union of the sets of vertices and the disjoint union of the sets of edges.

Figure fig. 1 shows $G \amalg H$, with $G = (\{1,2,3,4\}, \{(12),(32),(34)\}, s, t)$ and $H = (\{2,3\}, \{(23), s', t'\})$.

Definition 2.2 A path in a graph G is a sequence of edges $e_1 e_2 \dots e_n$ such that for all $i \in \{1, \dots, n-1\}$, $s_G(e_{i+1}) = t_G(e_i)$. A path $e_1 e_2 \dots e_n$ in $G \amalg H$ is said to be alternating between G and H when for all $i \in \{1, \dots n-1\}$, $e_i \in E_G$ if and only if $e_{i+1} \in E_H$.

The source $s_G(\pi)$ (resp. the target $t_G(\pi)$) of the path π is defined as $s_G(e_1)$ (resp. $t_G(e_n)$). A loop is a path p such that $s(p) = t(p)$, and finally a *closed path* (originally circuits) is an equivalence class of loops under cyclic permutations of the edges it is composed of.

We write $\mathrm{Paths}_{X \to Y}(G)$ the set of paths p in G such that $s(p) \in X, t(p) \in Y$, $\mathrm{AltP}_{X \to Y}(G, H)$ the set of alternating paths between G and H, and $\mathrm{C}(G, H)$ the set of closed alternating paths between G and H.

We sometimes associate colors to graphs, as in fig. 1, alternated paths then appear as color-alternating.

Definition 2.3 Given two interaction graphs G, H, their *execution* is the graph $G :: H$, with $V_{G::H} := V_G \ominus V_H = V_G \cup V_H \setminus V_G \cap V_H$ (symmetric difference), and $E_{G::H} := \mathrm{AltP}_{V_{G::H} \to V_{G::H}}(G, H)$.

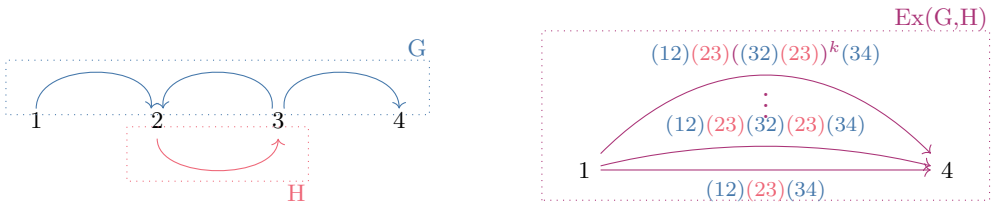


Fig. 1. Two graphs and their execution

Example 2.4 In Figure 1, (12)(23)(34) is an alternated path, (23)(32) is an alternated loop, and (23)(32) = (32)(23) as an alternated closed path. The right-hand picture shows the execution, where regular expressions are used to describe paths.

One fundamental result about execution is that it is associative (under a mild assumption).

Theorem 2.5 *Given three graphs G, H, J with $V_G \cap V_H \cap V_J = \emptyset$, then $G :: (H :: J) = (G :: H) :: J$.*

To make a model of Linear Logic out of interaction graphs, one then defines an orthogonality relation, which relates to the long-trip correctness criterion [25]. The latter being defined as the existence of exactly one closed path between two graphs, closed paths are of great importance in the study of interaction graphs. This motivates the following definition.

Definition 2.6 The measure of the interaction between G and H is defined as $[\![G, H]\!] := \sum_{\pi \in \mathrm{C}(G,H)} \pi$, where we use a formal sum notation for sets.

On top of the associativity of execution, the set of closed paths and execution satisfy a compatibility property called the *trefoil property* [27], or sometimes the *2-cocycle condition* [32].

Theorem 2.7 *If F, G, H are interaction graphs such that $V_F \cap V_G \cap V_H = \emptyset$,*

$$\begin{aligned} \mathrm{C}(F, G, H) &= \mathrm{C}(F, (G :: H)) \sqcup \mathrm{C}(G, H) \\ &= \mathrm{C}(H, (F :: G)) \sqcup \mathrm{C}(F, G) \\ &= \mathrm{C}(G, (H :: F)) \sqcup \mathrm{C}(H, F). \end{aligned}$$

This can be expressed with the measure as follows:

$$[\![F, (G :: H)]\!] + [\![G, H]\!] = [\![G, (H :: F)]\!] + [\![H, F]\!] = [\![H, (F :: G)]\!] + [\![F, G]\!].$$

In the specific case in which $V_G \cap V_H = \emptyset$, this property becomes the so-called *adjunction* property usually established in geometry of interaction models of linear logic.

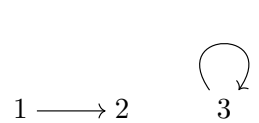
Corollary 2.8 (Adjunction) *If F, G, H are interaction graphs such that $V_G \cap V_H = \emptyset$,*

$$\mathrm{C}(F, (G \sqcup H)) = \mathrm{C}((F :: G), H) \sqcup \mathrm{C}(F, G).$$

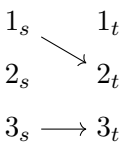
Written with notation of linear logic, one would want an adjunction between $\otimes$ and $\multimap$, that is that when measuring a graph $F : A \otimes B \to C$ against $G \otimes H : A \otimes B$, one should get the same measure as when measuring $F :: G : B \to C$ against $H : B$. But here it is not the case, because of the presence of the term $\mathrm{C}(F, G)$ in the adjunction 2.8. Indeed, when computing $F :: G$, the closed paths internal to the interaction between F and G disappear. This will be explained in more details in the next section.

3 Interaction Graph as a Traced Monoidal Category

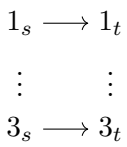
We will first establish that Seiller's model of Interaction Graphs [25] fits into the categorical framework of Geometry of Interaction, as a Traced Monoidal Category [19], which is one of the basic ingredients to make a (categorical) model of Geometry of Interaction [16].



(a) The graph G located on 1,2,3

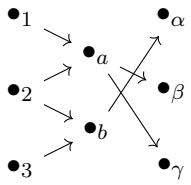


(b) It's obip representation $(\!|G|\!)$

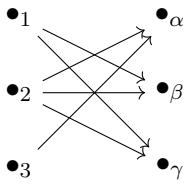


(c) The identity obip on $\{1, 2, 3\}$

Fig. 2. Interaction graphs and obips



(a) Two morphisms in **ioGrph**



(b) Their composition, ie paths of length 2

Notation 3.1 Given a set X, we will use the notation X_s to denote copies of X whose elements have been indexed by s, like x_s. In particular, we will use the sets X_s and X_t to create "source" and "target" copies of a set X. Indices enforce that these copies are disjoint, and provide canonical injections $i_s/i_t : X \to X_s \sqcup X_t$.

We will now assume that our graphs have edges with names taken from a free monoid. We can thus concatenate two names to make a new name. This provides a natural name to give to an arrow obtained as a contraction of a path: $(a_1, \dots a_n) \to a_1 \dots a_n$. This choice is a trick to avoid considering a quotient: defining a composition using pairs makes it not associative: $(a, (b, c)) \neq ((a, b), c)$ and would thus either require a quotient, or to work in a 2-category with explicit renaming of edges.

Definition 3.2 An oriented bipartite graph (obip) is a graph $G = (V_s \sqcup V_t, E, s, t)$ where for all $e \in E, s(e) \in V_s, t(e) \in V_t$. We represent these with V_s on the left (source) and V_t on the right (target).

A directed graph $G = (V, E, s, t)$ can be represented by the bipartite graph $(\!|G|\!) = (V_s \sqcup V_t, E, s', t')$ which has as vertices the disjoint union of two copies of V, and where the source and target functions are defined using the canonical injections i_s/i_t: $s'(e) := i_s(s(e))$ and $t'(e) := i_t(t(e))$.

Example 3.3 In fig. 2, we show an example of an interaction graph and its obip representation, as well as the natural representation of the identity obip. Notice how we use the previously defined notation, calling X_s and X_t the source and target set of the identity.

What is convenient with such a representation is that when glued together, paths are alternated by nature: to go from left to right, one has to take exactly one arrow from G and one from H. This naturally leads to the definition of the following category.

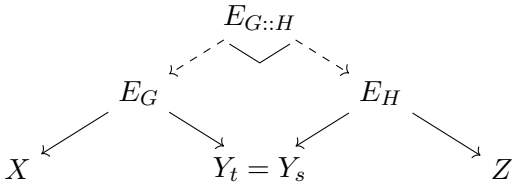
Definition 3.4 We define the category **ioGrph** of input/output graphs as the category with arbitrary sets as objects, and morphisms $G : A \to B$ the obip graphs whose edges have their source in A_s and target in B_t. The composition of $G \in \mathbf{ioGrph}(A, B)$ and $H \in \mathbf{ioGrph}(B, C)$ is is the graph $G :: H$ obtained by glueing G and H along B, and then computing the paths of length 2. Formally, it is the graph $(\!|G :: H|\!) = ((A_s \sqcup B_t), E', s', t')$ with $E' := \{ab \mid a \in E_G, b \in E_H, t_G(a) = s_H(b)\}$, $s'(ab) = s_G(a)$ and $t(ab) = t_H(b)$. The identity on X, written $1_X : X_s \to X_r$ is the graph $(X_s \sqcup X_t, X, s, t)$ with $s(x) = x_s, t(x) = x_t$.

Example 3.5 The sequence of morphisms $\{1, 2, 3\} \to \{a, b\}$ and $\{a, b\} \to \{\alpha, \beta, \gamma\}$ shown on the left side of Figure 3.5, once composed, give the graph on the right hand side.

Remark 3.6 Let us stress that **ioGrph** does not have any graphs without I/O other than the empty graph: $\mathbf{ioGrph}(A, \emptyset) = \mathbf{ioGrph}(\emptyset, B) = \{\emptyset\}$.

Remark 3.7 The category **ioGrph** is actually almost a "span category": it is similar to the 1-truncation (also called decategorification or homotopy category) of **Span**(**Set**). Indeed, an obip graph can be repre-

sented as a span with disjoint feet (so not exactly a span category). Composition is then given by pullback, creating an arrow for every pair of "composable arrows".



Remark 3.8 The "almost" in "almost a span" is the reason why dealing with interaction graph categorically can be really bureaucratic. Indeed, the composition is defined by taking the pullback on $C_t \simeq C_s$, inducing a need to use this identification (called a delocation) before performing the actual glueing. Delocations are thus used everywhere. The notation of writing s/t as indices, such as in C_s instead of just C, allows to have disjoint feet, and we put under the rug the delocations by pretending $C_s = C = C_t$. Note also that the result of composition still has disjoint feet for the same reason: L_s and R_t are disjoint because their elements have disjoint subscripts. We refer to this problem as *the first problem of locativity*.

Proposition 3.9 *The category* $(\imath o\mathbf{Grph}, \sqcup, I)$ *is monoidal. We will thus write* $\otimes$ *for* $\sqcup$ *in the following.*

The category **ioGrph** can be seen as a form of "proof-relevant" version of the category $\mathbf{Rel}_+$, the categories of relation, which is itself a generalisation of the category **PartFunc** of partial functions. These are well-known models of GoI, see [16].

Of course, we are interested in computing all alternated paths between G and H, not only those of lengths 2. Thus, after glueing G and H together and taking a path of length 2, we need to be allowed to go back at the beginning of G and go on. This is formalized by the concept of categorical trace [19].

The category **ioGrph** can indeed additionally be equipped with a trace, to make it a model of GoI. This operator will compute the paths in a graph where we can loop back to the beginning. There are two very similar constructions of such a graph, that can be used somehow interchangeably because the paths they induce do not differ. To picture these two constructions, look at fig. 4b. The first construction, $\circlearrowleft$ (feed) is the one adding the extra arrows that are seen going backwards. The second construction, $\square$ (fuse), is the same but contracting these arrows: we fuse their endpoints together directly.

Definition 3.10 Given an obip graph $G : I \otimes X_l \to O \otimes X_r$, let u be the forgetful function: $u(x_s) = u(x_l) = x$ and $u(v) = v$ otherwise, we define the graph $\square_X(G)$ by $V_{\square_X(G)} = I \sqcup O \sqcup X$, $E_{\square_X(G)} = E_G$, and

$$s_{\square_X(G)}(e) = u(s_G(e)), \qquad t_{\square_X(G)}(e) = u(t_G(e))$$

Definition 3.11 Given an opib graph $G : I \otimes X_l \to O \otimes X_r$, we define the graph $\circlearrowleft_X(G)$ (read as "feed X G")by $V_{\circlearrowleft_X(G)} = V_G$, $E_{G^{\circlearrowleft X}} = E_G \sqcup E_{1_X}$ and

$$s_{\circlearrowleft_X(G)}(e) = \begin{cases} s_G(e) & \text{if } e \in E_G \\ t(e) & \text{if } e \in E_{1_X} \end{cases}, \qquad t_{\circlearrowleft_X(G)}(e) = \begin{cases} t_G(e) & \text{if } e \in E_G \\ s(e) & \text{if } e \in E_{1_X} \end{cases}$$

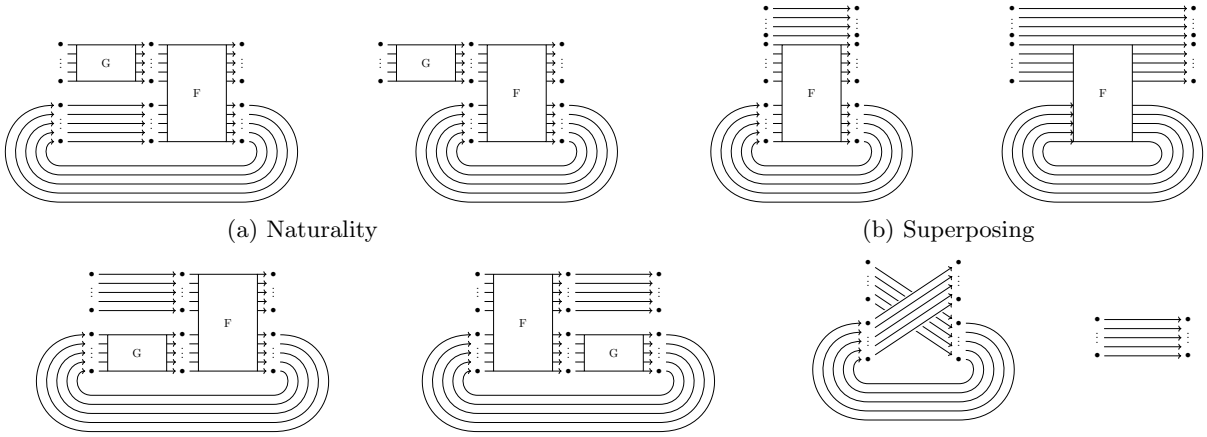
Proposition 3.12 $\forall i, o \in I, O : \mathrm{Paths}_{i\to o}(\square_X(G)) = \mathrm{Paths}_{i\to o}(\circlearrowleft_X(G))$.

From this proposition we know these two constructions are not really different from the input/output point of view, and so can both be used to define the same trace. Fuse ($\square$) is technically simpler to use in proofs, but Feed ($\circlearrowleft$) makes for much nicer drawings and we will use both interchangeably.

We can now define a categorical trace in **ioGrph** (main axioms are recalled in Figure 4). The proof is in the appendix.

Theorem 3.13 (ioGrph is Traced) *Given* $G \in \imath o\mathbf{Grph}(I \otimes X, O \otimes X)$*, we define* $\mathrm{Tr}^X_{I,O}(G)$ *by:*

$$V_{\mathrm{Tr}^X_{I,O}(G)} = A \sqcup B, \qquad E_{\mathrm{Tr}^X_{I,O}(G)} = \bigcup_{i\in I, o\in O} \mathrm{Paths}_{i\to o}(\square_X(G))$$



(a) Naturality (b) Superposing

(c) Dinaturality (d) Yanking

Fig. 4. Some axioms for the trace

with the source and target maps defined by the lifting to paths of the source/target of edges. The category $\imath o$**Grph** *endowed with this operation is a traced monoidal category.*

Note that something strange is at play: the computation of paths, which is the core of the execution formula, actually appears twice in these constructions: it appears in the definition of the composition :: in $\imath o$**Grph**, as we compute paths of lengths 2, and it appears again in the definition of $\mathrm{Tr}^X_{A,B}(G)$. These two operations of glueing and then computing the paths (that we name here saturation, as it is a similar phenomenon to the computation of saturated diagrams in [9]) are entangled together in the definition. One could aim to disentangle the two: to have glueing as a primitive operation, and saturation done once and for all on the final graph. This will be done in the next section.

Gathering the results we have established so far, we can now state the following theorem.

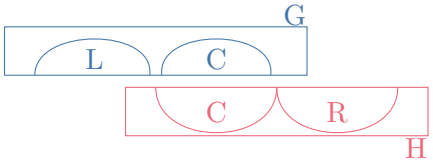
Theorem 3.14 *The category* $(\imath o\mathbf{Grph}, \sqcup, \sigma, \mathrm{Tr}^-_{-,-}(-))$ *is a (symmetric) traced monoidal category.*

From all this structure, we can define the category where interaction graphs can be interpreted, using the so-called $\mathbf{Int}(-)$ construction [16] (see annex for a reminder section A).

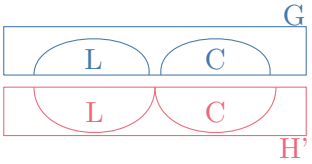
Definition 3.15 The category **iG** of interaction graphs is defined as $\mathbf{Int}(\imath o\mathbf{Grph})$.

This is the right category to interpret our programs: take two interaction graphs G with $V_G = L \sqcup C$ and H with $V_H = C \sqcup R$. Then we can see G as an obip, ie a morphism $(\!|G|\!) : L \otimes C \to L \otimes C$ in $\imath o$**Grph**, hence as a morphism $iG : L \to C$ in **iG**. Similarly H can be seen as a $iH : C \to R$. One can check that we have $i(G :: H) = iG :: iH$, where the :: on the left is the set-theoretic composition of definition 2.3, and the :: on the right the categorical composition in **iG**.

Remark 3.16 Like every categorical model dealing with locativity, these models suffer from the fact that category theory is oriented and typed. Take a graph G on $L \sqcup C$. It can be embedded in the category in multiple ways, depending on wether we want it to interact with an H on $C \sqcup R$ or an H' on $L \sqcup C$:

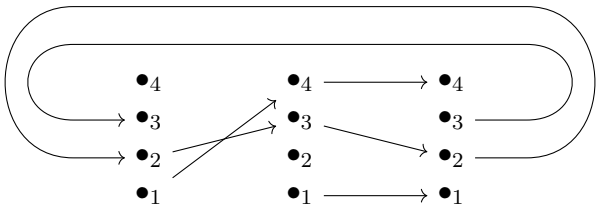


(a) G seen as a morphism $L \to C$

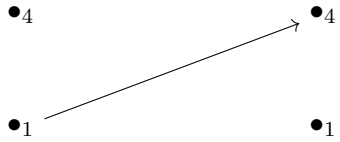


(b) G seen as a morphism $\emptyset \to L \sqcup C$

Thus there are many morphisms morally corresponding to the same graph, depending on what interface we want the graph to have. This change of interface is done categorically through the use of the adjunction coming from the closure: $\mathbf{iG}(L, C) = \mathbf{iG}(L \sqcup \emptyset, C) \simeq \mathbf{iG}(\emptyset, L \multimap C)$. Since the category is compact closed (result of the **Int**-construction), $\multimap$ and $\sqcup$ coincide, and we get the desired result. There is thus an extensive



(a) A composition of two interaction graph with trace, as in the **Int**-construction



(b) The execution of said graph, internal closed paths have disappeared

Fig. 6. Disappearing loops

and unnatural use of locations in order to transfer in and out of the center C and encode what would be a simple execution of three graphs $F :: G :: H$ in the original set-theoretical/location-based model.

As of today, there is no known way to fix this or make it simpler. This is referred to as *the problem of locativity* in [23], although it could be argued that *category theory is the problem*, in the sense that it is not the right tool to study these objects. We will refer to this as *the second problem of locativity*.

Unfortunately, some topological information is lost during composition in this category, just as is the case in the traditional "set-theoretic" presentation of interaction graphs, as hinted in section 2. This is problematic if one wants to make a model of Linear Logic, as it is lacking any interesting scalars $\mathbf{iG}(\emptyset,\emptyset) = \{1_\emptyset\}$ (this is a consequence of remark 3.6). But scalars are necessary to create a focus, an object used in the realisability construction of linear logic, which would categorically correspond to the process of (focussed) double gluing [18]. Computationally, this means there is no program corresponding to a "normal form of a good execution". We now explain what kind of information is lost and why exactly.

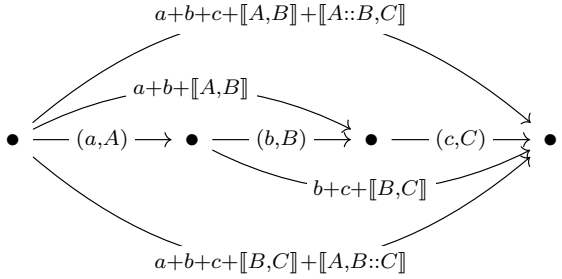
Example 3.17 The definition of trace uses $\mathrm{Paths}_{I\to O}(G)$, and thus all internal closed paths disappear (that is paths outside of the "connected component" of the elements of I, O). See for example the closed path $2 \to 3 \to 2$ in Figure 6 where we trace along $2, 3$.

To palliate to this problem, Girard and then Seiller introduced the concept of *wager*, an element in a monoid which intuitively counts the number of internal closed paths. By adjoining one to every program, one can get back the missing information. This new model of computation form a sister category to **iG**, with a richer set of scalars: the category of projects. This construction, reminiscent of [20], also relates to the nucleus of enriched adjunctions [11].

Definition 3.18 The category **Project** is defined as having for objects arbitrary sets, for morphisms $X \to Y$ pairs (A, a) where $a \in \Omega$ and $A \in \mathbf{iG}(X, Y)$. The composition of two morphisms (A, a) and (B, b) is given by $(A :: B, a + b + [\![A, B]\!]_\Omega)$.

The projection $\mathcal{U} : (A, b) \to A$ is a functor, which allows to see projects as a generalisation of interaction graphs. Beware, for the function $A \to (A, 0)$ of type $\mathbf{iG} \hookrightarrow \mathbf{Project}$ is *not a functor*, precisely because of the fact that there can be non-zero measures of the interaction of two graphs.

Remark 3.19 Composition has to be associative for **Project** to form a category. Checking associativity means comparing the top and bottom arrow in the following diagram:



(a) A quiver with the input/output property on I/O

(b) A representation of such graphs as a "blackbox"

Fig. 7. Examples of quivers

These two are equal precisely because of the *Trefoil Property* (Theorem 2.7). Notice also how it uses only 2 sides of the threefold equality stated in the trefoil property. This is related to remark 3.

This trefoil property can be seen abstractly as a statement of the measure being a 2-cocycle (this is a cohomological notion), and the reconstruction of the category **Project** is a classical construction in group cohomology theory, see [1] for the definition of 2-cocycle and [32] for some reference on this fact.

It thus seems that the category **iG** has "lost" some data compared to **Project**, hence the lack of scalars, and that this data was somehow recovered using the wager construction. But why lose and then recover the data? Is there a way to avoid losing this data in the first place? This phenomenon of losing data is something that does not happen in another category that has some similarities to **ioGrph**, the category of cobordisms, which is our original inspiration for this work and discussed in section B.

4 Preserving the Topology

Inspired by the category of cobordisms, we will argue that the "proper" way to compose interaction graphs to avoid forgetting the internal loops should be to compose them purely by glueing (pushout, cospan) and not by saturation/computing the paths (pullback, span). We will now use an alternative name for graphs for reasons that will become clearer later on.

Definition 4.1 A *quiver* is an oriented multi-graph. The category **Quiv** of quivers and quivers homomorphisms admits pushouts.

The intention behind the name quiver usually being that it is *the underlying graph of a category*. Glueing two obip quivers together does not produce an obip quiver, since there are 3 sets of vertices in the resulting object. We generalise the notion of obip to get a notion that is stable by glueing.

Definition 4.2 [Input/Output property] A quiver G is said to have the input/output property relatively to two sets $I \cap O = \emptyset, I \cup O \subseteq V_G$, when for every $e \in E_G$, $t(e) \notin I$ and $s(e) \notin O$.

Unfortunately, triplets (Q, I, O) with the input/output property, composed by glueing *do not form a category* as there is no identity. Intuitively, identities should be the same 1_X graphs as in **ioGrph**, but this morphism now no longer acts as an identity: it adds an extra set of vertices X to the graph, which do not disappear as in the previous case, and remain in the morphism resulting from the composition. We must therefore introduce a slightly weaker notion to capture the algebraic properties of our objects.

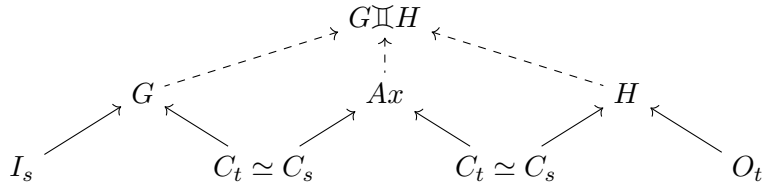
Definition 4.3 A semi-category (abbreviated as scategory) $\mathbf{C}$ is given by the same data as a category but without the requirement of having identities. Formally, it is defined as a set of objects O, sets of morphisms $\mathbf{C}(A, B)$ for all $A, B \in O$, and a composition $\mathbf{C}(A, B) \times \mathbf{C}(B, C) \to \mathbf{C}(A, C)$ which is associative.

A *semi-functor* is a function F, defined on objects and morphisms, such that $F(a; b) = F(a); F(b)$ without requirements on identities. See [17] for an exemple of use in computer science.

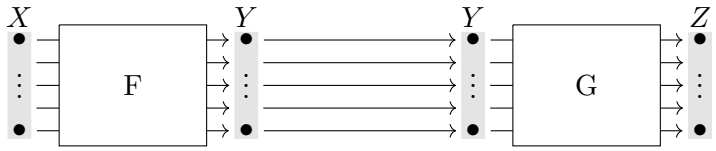
We can now define the structure capturing our notion of input/output quivers.

Definition 4.4 We define the *scategory* **ioQuiv**, whose objects are arbitrary sets, and morphisms $G : I \to O$ are quivers with the input/output property on I/O. The composition of $G : I \to C$ and $H : C \to O$, that we write $G \amalg H$ (read "G glue H") is the graph obtained by glueing G and H on the vertices in C.

Formally, this is defines as the following pushout diagram, in which Ax is just the identity graph id_C in **ioGrph**:

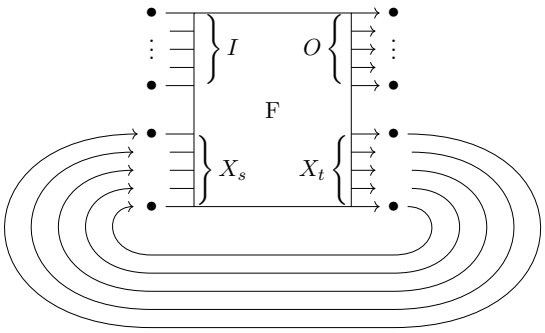


Remark 4.5 We could have chosen as composition the pushout directly, without Ax, which would impose to use fuse ($\square$) as a trace later on. We chosen to consider the additional Ax, and thus will take feed ($\circlearrowleft$). Here, since we are not computing the paths anymore, the choice fuse/feed do matter, and would give rise to two different (but very similar) scategories. Graphically the composition is as follows:



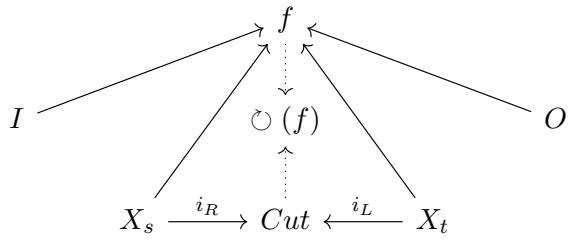
We will now explain how this scategory is almost traced monoidal (adapting straightforwardly the definitions from categories to scategories).

Definition 4.6 Given a morphism $f : I \otimes X \to O \otimes X$ in **ioQuiv**, we define $\circlearrowleft(f)$ as the morphism given by the expected drawing:

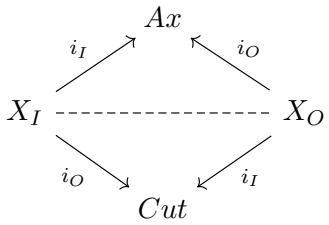


Notice that this is indeed a morphism in **ioQuiv** of type $I \to O$, as there are no more requirements on input/output on the X part, which is now invisible from the outside.

Remark 4.7 This "trace", similarly to the composition in the scategory, is also obtained as a pushout, between f and Cut, which is somehow symmetric to the pushout defining composition, taking Ax as the identity of **ioGrph**, Cut being just syntactic sugar for Ax with flipped injections:

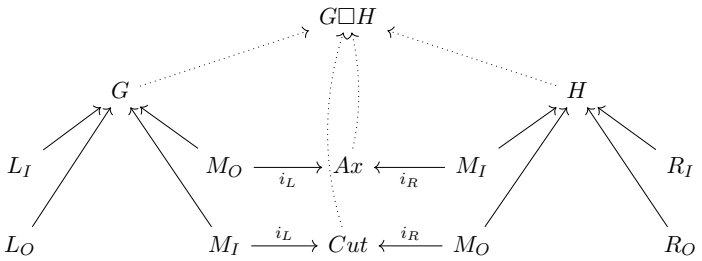


(a) The trace pushout



(b) Cut is the cospan symmetric to Ax

Note that Cut is *not a morphism in* **ioQuiv** since it has arrows from the output to the input, which invalidate both conditions of the input/output property. From both these constructions, an **Int**-like construction composition would then be of the following shape (still written $G \amalg H$), where L, M, R mean left/middle/right, I/O input/output.



This pushout with visible Ax/Cut alternation is the reason we chose feed instead of fuse as a trace.

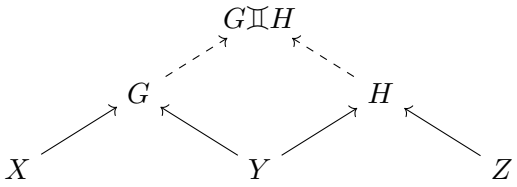
We can now easily check that this operation is the natural equivalent of a trace, in the setting of scategories. It satisfies all axioms expressible without identity morphisms, and therefore can be thought of as the scategory equivalent to a spatially traced category [33].

Proposition 4.8 *The $\circlearrowleft$ operator just defined satisfies all the axioms of a trace (imported verbatim in the setting of scategories), except for the axiom of Yanking.*

In fact, the Yanking axiom *cannot even be stated* for there are no identities. However, the morphism 1_X still somehow plays a role similar to the identity, and will become an identity in later constructions.

Since we compose morphisms by pushouts, this category is very similar to a category of cospans. It is close to the theory of structured cospans [4], where the feet I, O would be object of **Set** injected as discrete graphs into **Quiv**. There are however two differences: we have to perform the delocalisation trick $C_t \simeq C_s$ to keep feet disjoint 3.8, and we can only inject the feet onto "boundaries" I/O.

From the cospan point of view, the usual composition of interaction graphs is done in three steps (we ignore the requirement for alternation, this could be done using the same trick as the **Int**-construction). First, we have a glueing step as in **ioQuiv**:



Second, we have a saturation step (dynamics): we compute all possible paths, which corresponds to taking a free construction over the resulting graph. Third, we have a "hiding" step in which we erase the internal machinery. Note that this last step is only necessary if one wants a *denotational* semantics.

Under this lens, what is usually referred to as "Hiding" is decomposed into two different steps. First, hiding the inside: the pushout hides Y, and the inner workings of the program become invisible from the outside, only the boundaries are visible. Second, erasing the inside: we remove the extra vertices added when glueing the boundaries. In particular, **ioQuiv** is almost a category, up to erasure of such vertices when glueing with an identity.

We can now pinpoint where things go wrong in the setting of interaction graphs, namely the erasure step. Because a loop needs a vertex to exist, inside loops are erased when their base vertices are removed. It is in the erasure that things go wrong in the setting of interaction graphs: because of their discreet nature, a loop needs a vertex to exist, and when such a vertex is erased then the loop disappears.

One can sum up these observations in the following way:

Execution transforms coSpans into Spans, by Saturation and Hiding

We can consider two opposite ways of solving the problem. The first is to avoid erasing the inside points. As stated above, we cannot work with categories anymore, and we are led work within a theory of scategories. We can then re-introduce hiding in a controlled fashion as 2-morphisms. This is done in [23].

The second direction is to use the "nice property" that continuous spaces, like cobordisms, have: the boundary points that are fused "melt" into the space: for example, when glueing two lines on an endpoint, the endpoint becomes part of the resulting line and is not distinguished anymore. Thus it is erased "for free". Moreover, spaces can exist even without distinguished points, in particular there can be spaces

(a) The oriented circle

(b) The two-arrows construction

Fig. 9. Examples of directed spaces

without boundaries, like the circle, hence no information disappears. This is the line of research pursued in the remaining sections.

5 Realising Interaction Graphs as Spaces

We will now associate to every graph a space called its *geometric realisation*. This almost corresponds to understanding a graph as the drawing we make of it. Not exactly though, because we will need to identify some spaces differing only by un-important deformations: it should not matter wether an edge is shorter or longer, what really matters is the geometry of the space.

Since our graphs are oriented, and therefore our edges carry a notion of direction, considering standard topological spaces is not possible and we will have to exploit the notion of *directed space*. We will start with a quick review of the notion and basic properties, and refer to the literature for more details: the reference textbook [10] and/or Dubut's PhD thesis [8].

There are in fact two directions one can follow in order to solve the problem of the wager. First, we can compose graphs (seen as spaces) by glueing them together on their boundaries, in the spirit of **ioQuiv**. Alternatively, we could see the programs not as graphs or spaces, but as their induced categories of paths, with composition given by pushout as well. In this second approach, execution (the computation of paths) would be built in the composition, since the composition of a category is a total function, and thus the pushout of two categories, being a category itself, would have all possible compositions of morphisms of both, ie all paths.

Both approaches are consistent with the other: execution on a space correspond to the computing of paths in the space S, that is determining the fundamental category $d\Pi_1(S)$. Van Kampen's theorem will ultimately ensure that they lead to the same result.

5.1 Geometric Realisation of Interaction Graphs

Notation 5.1 We will denote by I the unit interval of $\mathbb{R}$, $s(I) = 0 \in I$ and $t(I) = 1 \in I$.

Definition 5.2 A directed framework (*df*) on a topological space X is a set $dX \subseteq \mathbf{Top}(I, X)$ of *directed path of X*, or *dipaths*, satisfying the following axioms:

- (Constant Path) If f constant, then $f \in dX$.
- (Stability by Reparametrization) For all increasing $r : I \to I$, $r; f \in dX$.
- (Stability by concatenation) If $f(1) = g(0)$ (the boundaries match) then $f; g \in dX$.

We write dI for I equipped with the directed framework dI of increasing functions.

Definition 5.3 A directed space (dspace) is a pair (X, dX) with X a topological space and dX a directed framework on that space. A directed map (dmap) $f : (X, dX) \to (Y, dY)$ is a continuous function $X \to Y$ such that $f \circ dX \subseteq dY$.

We write **dTop** the category whose objects are directed spaces and morphisms are dmaps. Note in this context the natural notion of isomorphism is stricter than in the undirected case (homeomorphism)

In the following, we will denote by $dX(a, b) \subseteq dX$ the set of dipaths from a to b. Since we mostly consider directed spaces, we will often omit the dX part.

Example 5.4 As a first example, the oriented circle is given by glueing the endpoints i and o of the space dI. Formally it is the space (S^1, dS) with dS the maps $f : I \to S$ such that, when represented in polar coordinates as $f : t \to e^{i.2\pi.g(t)}$ have g increasing (we only go counter-clockwise). As a second example, the two arrows construction is given by glueing two oriented intervals dI on their endpoints i and o. Notice there are two paths $i \to o$ and no paths $o \to i$.

Both of these spaces are pictured in fig. 9. If these two were regular topological spaces, they would be homeomorphic but the orientation prevents them from being dihomeomorphic.

As stated earlier, we will need to consider our objects up to a quotient (a deformation). We will also have to consider multiple distinguished points (Input/Output). Given two object A, B with the same set of distinguished points (I, O), we will say a map $A \to B$ is boundary-preserving when $\forall x \in I, O, f(x) = x$. This will induce a stronger notion of isomorphism. For example, we will say that two d-space X, Y are equal up to boundary-preserving isomorphism when they are isomorphic in the sense of directed maps with two maps that are boundary-preserving, and we will write it $X \simeq Y$. We will have the same notion for categories.

The following standard property will also be essential, as it will allow us to glue dspaces.

Proposition 5.5 *The category* **dTop** *is cocomplete. In particular, it has pushouts.*

We are now ready to define the *geometric realization* of a graph, that is a dspace representation of it.

Definition 5.6 Given X a dspace, and $x, y \in X$, the *amalmagation* $X[x = y]$ is the quotient space of X where x and y are identified, and the directed path are (up to reparametrization) the finite sequences of $f_i : a_i \to b_i$ with $a_i, b_i \in \{x, y\}$ except (possibly) for a_0, b_n. This can be extended easily to an amalgamation $X[S]$ where S is a set of equations between points such as $x = y$.

Proposition 5.7 *Given a graph $G = (V, E, s, t)$, we define the dspace $\mathcal{R}(G)$ as the amalgamation*

$$V \bigsqcup_{e \in E} I_e[\{a = b \mid a \in V, b = s(I_e) \text{ with } s(e) = a, \text{or } b = t(I_e) \text{with } t(e) = a\}],$$

where $(I_e)_{e \in E}$ is a family of copies of the interval I, and the directed framework is the natural one. This function $\mathcal{R}$ can moreover be extended into a functor $\mathcal{R} : \mathbf{Quiv} \to \mathbf{dTop}$ with obvious effect on morphism.

Since we compose our graphs by glueing the together, it is important that $\mathcal{R}$ preserves this operation.

Proposition 5.8 *The functor $\mathcal{R}$ preserves colimits (it is left adjoint to a functor called the Nerve). Since a pushout is a special case of colimit, $\mathcal{R}$ preserves pushouts.*

We will note $G \Vdash S$ to say that $S = \mathcal{R}(G)$, taking inspiration from the notations of realisability theory [2], as G is like a syntactic description of the "real space" S.

We are now ready to define the category that we have been searching for. Note that it is *a category* not just a *scategory*. We obtained the identity back.

Definition 5.9 We define the category **ioSpace** of geometric realization of quivers, composed by glueing. Objects are arbitrary sets (of points), and morphisms $I \to O$ are (equivalence classes up to boundary-preserving equivalence of) directed spaces S such that there exists a directed graph $G : I \to O$ in **ioQuiv** with $G \Vdash S$. The composition of $S_1 : X \to Y$ and $S_2 : Y \to Z$, written $S_1 \between S_2$, is defined as $\mathcal{R}(G_1 \between G_2)$ which is well defined since $\mathcal{R}(-)$ preserves pushouts. Finally, identities are $\mathcal{R}(1_X) : X \to X$.

We now need to prove that this category is sufficient to be a model of Geometry of Interaction, which is fairly easy. Since isomorphisms preserve the boundaries in X and $\circlearrowleft$ just glues said boundaries, it is easily lifted, giving the following proposition.

Proposition 5.10 *If $\mathcal{R}(G) \simeq \mathcal{R}(H)$, then $\mathcal{R}(\circlearrowleft(G)) \simeq \mathcal{R}(\circlearrowleft(H))$.*

Thus the geometric realization of a $\circlearrowleft(G)$ does not depend on the "geometric" choice of G, and this notion can be lifted to spaces. We can import the $\circlearrowleft$ operator of **ioQuiv** in **ioSpace** by defining, for $G \Vdash S : X \otimes I \to X \otimes O$, $\circlearrowleft(S) = \mathcal{R}(\circlearrowleft(G))$.

Proposition 5.11 *The category $(\textbf{ioSpace}, \sqcup, \emptyset, \circlearrowleft)$ is traced monoidal. We will once again write $\sqcup$ as $\otimes$.*

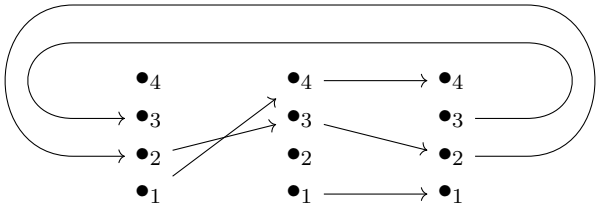
Proof. The only axiom which failed in the case of **ioQuiv** was the Yanking axiom $\mathrm{Tr}^X_{X,X}(\sigma) = 1_X$. But it is visually clear that $\mathcal{R}(\mathrm{Tr}^X_{X,X}(\sigma)) \simeq_{dhom} \mathcal{R}(1_X)$ in the picture fig. 4d. □

Remark 5.12 The $\circlearrowleft$ operator is really reminiscent of the geometric realization of a while loop in [10].

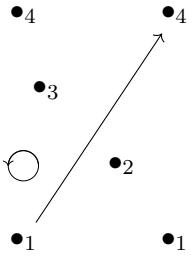
From that we can define the category where we interpret interaction graphs as spaces.

Definition 5.13 We define the category of interaction spaces **iS** as **Int**(**ioSpace**).

From this we could then reiterate the theory of interaction graphs. But here, we keep the extra information that was previously lost. Because **ioQuiv** says nothing about what happens inside the graphs, many scalars are permitted, such as a vertices with two loops.



(a) The same composition than 3.17, now is a morphism $\{1,4\}_s \to \{1,4\}_t$ in what **Int**(**ioQuiv**) would be



(b) The geometric realisation of the morphism

Remark 5.14 This paper only deals with the qualitative type of interaction graphs, but there exists a quantitative version [26], [31] which uses graphs with weighted edges. It is an open problem how one would adapt these considerations here: normally, it would not be possible to associate a weight to "every edge", since there are no longer edges, and the naive idea of encoding the weight in the geometry – such as considering the length of paths, does not work. For now, the simplest way might be, since we defined **ioSpace** not intrisically but as a realization of **ioQuiv**, to assign weights to the edges in a $G \Vdash S$ in a consistent way depending on the choice of G. But even if possible, this would not generalise to a model like **ioSpace** defined intrinsically as geometric objects with boundaries.

Now we turn our attention to relating this new category to the previously defined ones. For that, we will need to compute paths in the constructed space, that is construct the fundamental category $d\Pi_1$.

5.2 Paths in the space

We will define the functor mapping a directed I/O space to its category of paths. But we do not want to consider all paths: for example, going faster or slower but doing the same overall travel should count as the same path. What we want is to consider paths up to dihomotopy (*directed homotopy*). We could also, instead of equating such paths, make a 2-category by adding 2-cells representing the dihomotopies, but this would make things more complicated.

In topology, it is possible to make the set of continuous maps from A to B into a topological space $A \to B$. We do it by considering the compact-open topology.

Definition 5.15 Let A, B be topological spaces, take $K \subseteq A$ compact, $X \subseteq B$ open, $V(K,X)$ is the set of functions f such that $f(K) \subseteq X$. The *compact open topology* is the topology on the space of functions, generated by taking the $V(K,X)$ as subbasis.

From that we can see the directed framework dX, which is a function space, as a space itself: the subspace of the space $\mathbf{Top}(I,X)$ equipped with the compact-open topology.

Definition 5.16 Given two paths $\gamma, \delta : x \to y$ in dX, a dihomotopy is a path $H : \gamma \to \delta$ in dX, that is, a continuous function $H : I \to dX$ such that $H(0) = \gamma, H(1) = \delta$.

Notice how this relates to the usual notion of homotopy: H is an homotopy in the usual sense, but such that all the paths in between γ and δ are directed. We can now define the *fundamental category*, the category whose morphisms are the paths inside the space.

Definition 5.17 Given a dspace S, we define its *Fundamental Category* $d\Pi_1(S)$ as follows. Objects are the points of the space S, and morphisms $x \to y$ are equivalence classes of paths $[\gamma] : x \to y$ up to

dihomotopy. Composition is given by the composition (or concatenation) of paths, and the identity is the trivial constant path at x. This extends to a functor $d\Pi_1 : \mathbf{dTop} \to \mathbf{Cat}$, by defining the action on dmaps as follows: $d\Pi_1(f)$ is a functor, with $d\Pi_1(f)(x) = f(x)$, $d\Pi_1(f)(\gamma : x \to y) = f \circ \gamma$.

If one wants to keep a "discrete" point of view, this is almost the right notion to replace the notion of graph: the length of edges does not matter because of the quotient by dihomotopy (in fact, the length is just the special case of reparametrization). The only problematic thing is that the points (objects) do matter in the fundamental category, and thus extending an edge gives new objects that do not really matter. The notion corresponding to an interaction graph should probably be a fundamental category *up to equivalence of categories*.

We remind the reader that **Cat** has all small colimits, and thus pushouts. We can thus extend the definition of the input/output property to categories and define a scategory similar to **ioQuiv**.

Definition 5.18 We define the scategory **ioCat** as having arbitrary sets as objects, morphisms $\mathbf{G} : I \to O$ are equivalence class up to boundary-preserving directed equivalence of categories [15] with the input/output property on I/O, $\mathbf{G}(I, O) = \emptyset = \mathbf{G}(O, X)$. The composition of $\mathbf{G} : I \to C$ and $\mathbf{H} : C \to O$, that we write $\mathbf{G} :: \mathbf{H}$ is obtained by the same pushout as **ioQuiv**.

Following what was done with quivers and their geometric realisations, we can directly extract from those the category of paths, or *fundamental category*. To a quiver in **Quiv** one can associate a category of its path, with a functor analogous to $d\Pi_1$.

Proposition 5.19 *There is an adjunction between the forgetful functor U and the free category (path category) construction $\mathcal{D}$ as follows:*

$$\mathbf{Quiv} \underset{\mathcal{U}}{\overset{\mathcal{D}}{\rightleftarrows}} \mathbf{Cat} \quad (\perp)$$

We use $\mathcal{D}$ to stand for "dynamics" (instead of the usual name, $\mathcal{F}$ree), for this functor brings dynamics inside the purely geometric notion of graph. This can be lifted as a semi-functor $\mathcal{D} : \mathit{io}\mathbf{Quiv} \to \mathit{io}\mathbf{Cat}$.

We could interpret directly interaction graphs in **ioCat** instead of **ioSpace**, which is very in the spirit of [29], [31] where edges are functions. Indeed, our edges would now *compose*, and we can impose equations on the composition of edges, while only the free paths exist in interaction graphs.

We can finaly express the link between our categories **ioCat** and **ioSpace**. The functor $d\Pi_1(-)$ preserves certain pushouts by Van Kampen's theorem. Moreover, it sends ([8], 4.2.1 corollary 3) dihomomorphic spaces to equivalent categories. From this follows the next result.

Proposition 5.20 (Van Kampen Theorem) *The functor $d\Pi_1 : \mathbf{dTop} \to \mathbf{Cat}$ can be lifted to a semi-functor $d\Pi_1 : \mathit{io}\mathbf{Space} \to \mathit{io}\mathbf{Cat}$.*

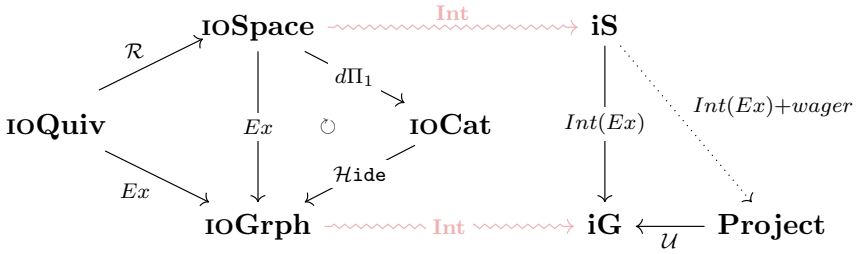
The paths $I \to O$ do not depend on the choice of representative of the equivalence class of categories, because we consider boundary-preserving equivalences.

Proposition 5.21 *Given two equivalent categories $\mathbf{C} \simeq \mathbf{D} : I \to O$ with a boundary-preserving equivalence, then $\mathbf{C}(I, O) \simeq \mathbf{D}(I, O)$.*

This allows us to extract just these paths, and close back the loop to our original category **ioGrph**.

Definition 5.22 There is a semi-functor $\mathcal{H}\mathtt{ide} : \mathbf{ioCat} \to \mathbf{ioGrph}$ defined by $\mathcal{H}\mathtt{ide}(A) = A$ on objects and $\mathcal{H}\mathtt{ide}(f : I \to O) = \mathrm{Hom}_f(I, O)$ seen as a obip graph on morphisms.

We sum up everything we discussed in the following diagramn where $Ex : \mathbf{ioSpace} \to \mathbf{ioGrph}$ is defined as the composition $d\Pi_1; \mathcal{H}\mathtt{ide}$.



One can see that the wager construction, which counts all the possible circuits, can be obtained by homological methods, giving a functor into **Project**. This is not completely surprising considering its origins is found in correctness criterions, which have already been reexpressed in the language of homology [24]. Notice that we considered paths up to dihomotopy, while the obtention of closed paths is done via homology. This begs the question as to whether homology or homotopy is the right tool to try and generalise this work to higher dimensions.

Another thing to note was that **ioGrph** allowed for double glueing, using a family of relations and not a focus. From what we have seen, it turns out the reason is that it is *the image* of a category with a focus. One might wonder if this is more general: can any category with a non-focussed orthogonality be seen as the projection of a larger category together with a focussed orthogonality?

We end this paper with one really important thing of note: the trefoil property (Theorem 2.7), which is the key to get the orthogonality and a model of linear logic, was about the fact that some set of cycles could be equated. This can now be seen a simple consequence of the fact that composition in **iS** is associative. Indeed, $\mathrm{C}(F,(G::H))\sqcup\mathrm{C}(G,H)=\mathrm{C}(H,(F::G))\sqcup\mathrm{C}(F,G)=\mathrm{C}(G,(H::F))\sqcup\mathrm{C}(H,F)$ because all these are ways of counting closed paths in the same geometric object $F\barwedge G\barwedge H$, which is well defined thanks to the associativity of $\barwedge$. Note also that once again, the problem of locativity (remark section 3) creeps up here: the real geometrical object is not a composition of morphism, but a glueing of the spaces F, G and H, for an associative and commutative glueing operation. It is not the case for the glueing composition $\barwedge$ in **ioQuiv** because of typing: if $F : A \to B, G : B \to C$ then $F\barwedge G$ is defined but not $G\barwedge F$.

6 Conclusion

Although mathematically demanding, we think this is a promising new approach to Geometry of Interaction seen from the categorical point of view. It would be quite valuable to be able to use tools coming from algebraic topology to get information on programs once interpreted inside interaction graphs (so on lambda terms for example). While the construction is here detailed for the particular simple case of interaction graphs, it already opens several directions for extension.

First, the geometric direction (**ioSpace**), to try and make new traced monoidal categories to generate models of GoI, based on more complicated topological objects. Graphs being one dimensional simplicial complexes, one might try to define higher dimensional interaction graphs, glued on their boundaries. More abstractly, there might be a general construction at work that can create models of GoI from cospan-like categories. If they have a notion of "Van Kampen", one might even be able to define computation (a normal form). As such, there are probable links with [21].

Second, the categorical direction (**ioCat**), which considers as primitive "programs" the path categories instead of just graphs. This gives extra power, for now morphism can be composed but equations can be imposed on morphisms. How do other models of GoI (notably, the extension of Interaction Graphs, such as the quantitative model) can be explained in these terms? One would expect the most general one, Graphings [29] to have objects with a richer structure (topological spaces) while retaining some of the fundamental constructions seen here.

One extra open question of particular interest is the link between directedness and undirectedness, which is not as easy to understand as one would expect, as explained at the end of the appendix section B.

Lastly, since game semantics [7] – especially AJM Games which are closely related to GoI [3] –, usually considers that execution consists in composition and hiding, it would be interesting to understand what becomes of our threefold decomposition as composition, saturation, and hiding.

References

[1] *2-cocycle for a group action - Groupprops.*
https://groupprops.subwiki.org/wiki/2-cocycle_for_a_group_action

[2] *Realizability* (2008), ISBN 978-0-444-51584-1.
https://shop.elsevier.com/books/realizability/van-oosten/978-0-444-51584-1

[3] Abramsky, S. and R. Jagadeesan, *Games and full completeness for multiplicative linear logic*, The Journal of Symbolic Logic **59**, pages 543–574 (1994). Publisher: Cambridge University Press.
https://doi.org/10.2307/2275407

[4] Baez, J. C. and K. Courser, *Structured Cospans* (2020). ArXiv:1911.04630 [math].
https://doi.org/10.48550/arXiv.1911.04630

[5] Brown, M., *Locally Flat Imbeddings of Topological Manifolds*, Annals of Mathematics **75**, pages 331–341 (1962), ISSN 0003-486X. Publisher: [Annals of Mathematics, Trustees of Princeton University on Behalf of the Annals of Mathematics, Mathematics Department, Princeton University].
https://doi.org/10.2307/1970177

[6] Brown, R., *Topology and Groupoids* .

[7] Clairambault, P., *Causal Investigations in Interactive Semantics*, Habilitation à diriger des recherches, Aix-Marseile Université (2024).
https://hal.science/tel-04523273

[8] Dubut, J., *Directed homotopy and homology theories for geometric models of true concurrency*, These de doctorat, Université Paris-Saclay (ComUE) (2017).
https://theses.fr/2017SACLN032

[9] Eng, B., *An exegesis of transcendental syntax : A journey into the logical machinery*, These de doctorat, Paris 13 (2023).
https://theses.fr/2023PA131018

[10] Fajstrup, L., E. Goubault, E. Haucourt, S. Mimram and M. Raussen, *Directed Algebraic Topology and Concurrency*, Springer International Publishing, Cham (2016), ISBN 978-3-319-15397-1 978-3-319-15398-8.
https://doi.org/10.1007/978-3-319-15398-8

[11] Gastaldi, J. L., S. Jarvis, T. Seiller and J. Terilla, *Linear realizability and structures in $\mathbb{R}$-enriched adjunctions* (2026). Preprint, available from the authors.

[12] Girard, J.-Y., *Linear logic*, Theoretical computer science **50**, pages 1–101 (1987). Publisher: Elsevier.
https://doi.org/10.1016/0304-3975(87)90045-4

[13] Girard, J.-Y., *Geometry of interaction I: interpretation of System F*, in: *Studies in Logic and the Foundations of Mathematics*, volume 127, pages 221–260, Elsevier (1989).
https://doi.org/10.1016/S0049-237X(08)70271-4

[14] Girard, J.-Y., *Geometry of interaction V: logic in the hyperfinite factor*, Theoretical Computer Science **412**, pages 1860–1883 (2011). Publisher: Elsevier.
https://doi.org/10.1016/j.tcs.2010.12.016

[15] Grandis, M., *Directed Algebraic Topology: Models of Non-Reversible Worlds*, Cambridge University Press, Cambridge (2009), ISBN 978-0-521-76036-2.

[16] Haghverdi, E. and P. Scott, *Geometry of Interaction and the Dynamics of Proof Reduction: A Tutorial*, in: B. Coecke, editor, *New Structures for Physics*, pages 357–417, Springer, Berlin, Heidelberg (2011), ISBN 978-3-642-12821-9.
https://doi.org/10.1007/978-3-642-12821-9_5

[17] Hayashi, S., *Adjunction of semifunctors: Categorical structures in nonextensional lambda calculus*, Theoretical Computer Science **41**, pages 95–104 (1985), ISSN 0304-3975.
https://doi.org/10.1016/0304-3975(85)90062-3

[18] Hyland, M. and A. Schalk, *Glueing and orthogonality for models of linear logic*, Theoretical Computer Science **294**, pages 183–231 (2003), ISSN 0304-3975.
https://doi.org/10.1016/S0304-3975(01)00241-9

[19] Joyal, A., R. Street and D. Verity, *Traced monoidal categories*, Mathematical Proceedings of the Cambridge Philosophical Society **119**, pages 447–468 (1996), ISSN 0305-0041, 1469-8064. Publisher: Cambridge University Press (CUP).
https://doi.org/10.1017/s0305004100074338

[20] Kelly, G. M. and M. L. Laplaza, *Coherence for compact closed categories*, Journal of Pure and Applied Algebra **19**, pages 193–213 (1980), ISSN 0022-4049.
https://doi.org/10.1016/0022-4049(80)90101-2

[21] Lack, S. and P. Sobociński, *Adhesive and quasiadhesive categories*, RAIRO - Theoretical Informatics and Applications **39**, pages 511–545 (2005), ISSN 0988-3754, 1290-385X.
https://doi.org/10.1051/ita:2005028

[22] Lucquin, A., L. Pellissier and T. Seiller, *Linear realisability and implicative algebras* (2026). Submitted.
https://hal.archives-ouvertes.fr/hal-05474801

[23] Maestracci, V., *Un petit pas vers une Logique Open Source*, PhD Thesis (2025).
2025AIXM0360/document

[24] Métayer, F., *Une étude homologique des réseaux*, These de doctorat, Paris 7 (1994).
https://theses.fr/1994PA077349

[25] Seiller, T., *Interaction graphs: Multiplicatives*, Annals of Pure and Applied Logic **163**, pages 1808–1837 (2012), ISSN 0168-0072.
https://doi.org/10.1016/j.apal.2012.04.005

[26] Seiller, T., *Logique dans le facteur hyperfini : Géométrie de l' interaction et complexité*, These de doctorat, Aix-Marseille (2012).
https://theses.fr/2012AIXM4064

[27] Seiller, T., *Interaction graphs: Additives*, Annals of Pure and Applied Logic **167**, pages 95–154 (2016), ISSN 0168-0072.
https://doi.org/10.1016/j.apal.2015.10.001

[28] Seiller, T., *Interaction Graphs: Full Linear Logic*, in: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '16, pages 427–436, Association for Computing Machinery, New York, NY, USA (2016), ISBN 978-1-4503-4391-6.
https://doi.org/10.1145/2933575.2934568

[29] Seiller, T., *Interaction graphs: Graphings*, Annals of Pure and Applied Logic **168**, pages 278–320 (2017), ISSN 0168-0072.
https://doi.org/10.1016/j.apal.2016.10.007

[30] Seiller, T., *Interaction Graphs: Exponentials*, Logical Methods in Computer Science **Volume 15, Issue 3** (2019), ISSN 1860-5974. Publisher: Episciences.org.
https://doi.org/10.23638/LMCS-15(3:25)2019

[31] Seiller, T., *Mathematical Informatics*, thesis, Université Sorbonne Paris Nord (2024).
https://theses.hal.science/tel-04616661

[32] Seiller, T., *Zeta Functions and the (Linear) Logic of Markov Processes*, Logical Methods in Computer Science **Volume 20, Issue 3**, page 10303 (2024), ISSN 1860-5974.
https://doi.org/10.46298/lmcs-20(3:18)2024

[33] Selinger, P., *A survey of graphical languages for monoidal categories*, volume 813, pages 289–355 (2010). ArXiv:0908.3347 [math].
https://doi.org/10.1007/978-3-642-12821-9_4

[34] Thom, R., *Quelques propriétés globales des variétés différentiables*, Commentarii Mathematici Helvetici **28**, pages 17–86 (1954), ISSN 1420-8946.
https://doi.org/10.1007/BF02566923

A Reminders on Int-Construction

We here do a quick "pedagogical" explanation for the **Int**-construction using intuitions on interaction graphs.

Imagine one wants to compute the alternating paths between two graphs G and H, as is done in the execution formula for interaction graphs 2.3. Computing the paths is simple: glue them together and then just go through edges sequentially (do a free category construction).

The problem is that we need to find a way to enforce alternation. The idea behind the Int-construction is to make a picture such that if you want to traverse it you have to first take an arrow in G then and only then can you take an arrow in H.

In more details, we unfold G and H and present them as bipartite graphs from $V_G \to V_G$ (resp, V_H), hence the definition of **ioGrph**.

We put the graphs together side by side, since the arrows are oriented from left to right in a bipartite graph, you have to first take an arrow in G then in H.

Because of the geometry of such a construction, paths cannot take an edge from the same graph twice in a row, and thus have to be alternated by construction. The trace is then used as a sort of "feedback", allowing to go back and continue the path to obtain paths of length greater than 2. Formally:

Definition A.1 [**Int**-construction] Given a category $\mathbf{C}$, we define the category $\mathbf{Int}(\mathbf{C})$ as follows:

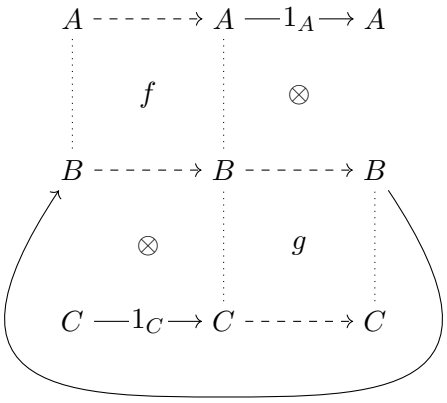
Objects: Pairs $(+A, -A)$ of objects of $\mathbf{C}$

Morphisms: $f : (+A, -A) \to (+B, -B)$ are morphisms in $\mathbf{C}$ of the type $+A \otimes -B \to -A \otimes +B$

Given $f : (+A, -A) \to (+B, -B)$ and $g : (+B, -B) \to (+C, -C)$, the composition is defined as first plugging f and g on $+B$ as follows:

$$+A \otimes -B \otimes C \xrightarrow{f \otimes 1_{-C}} -A \otimes +B \otimes -C \xrightarrow{1_{-A} \otimes g} -A \otimes -B \otimes +C$$

And then tracing over $-B$. Pictorially, this is what happens:



Paths go from left to right, except when looping. They are forced to alternate by construction.

Theorem A.2 *Given a traced monoidal category $(C, \otimes, I, \dots)$, $\mathbf{Int}(C)$ is the free compact closed category containing C.*
Moreover, the functorial embedding $\mathbf{C} \to \mathbf{Int}(\mathbf{C}) : A \to (A, I)$ is fully faithfull. See [19] and [16].

B A quick overview of Cobordisms and the origin of this paper

B.1 How cobordism solves the wager problem

Cobordisms were introduced by René Thom in 1954 [34] in the search of nice geometric invariants of spaces.

We recall that a smooth manifold is a topological space which is locally homeomorphic to a Euclidean space (so, a topological manifold) and such that the gluing functions which relate these Euclidean local charts to each other are smooth.
One can then glue together such manifolds M and N on a common boundary B by taking the disjoint union $M \sqcup N$ here, and quotienting the result by identifying the two copies of B. This gives another smooth manifold, equipped with the quotient topology.
We first present the category of cobordisms as a form of cospan. One should note it is *not a cospan category*, but a subcategory of one, because the foot are not arbitrary topological spaces, and because they need to be sent to boundaries.

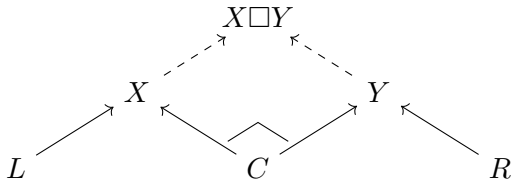
Definition B.1 The category $\mathbf{Cob}_{[n]}$ is defined as follows:

Objects: Closed smooth manifolds of dimension $n-1$.

Morphisms: The set $\mathbf{Cob}_{[n]}(L,C)$ of morphisms from L to C is the set of smooth manifolds with boundarys $\mathcal{M}$ of dimension n whose boundary $\partial\mathcal{M}$ is equal to $L \sqcup C$.

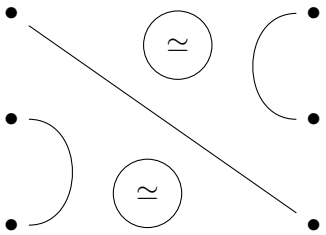
Composition: Composition is given by gluing cobordisms along their shared boundaries. Formally, given $\mathcal{M} \in \mathbf{Cob}_{[n]}(L,C)$ and $\mathcal{N} \in \mathbf{Cob}_{[n]}(C,R)$, the cobordism $\mathcal{M};\mathcal{N}$ is defined as the smooth manifold $(M \sqcup N)_{/\sim}$ where the quotient is defined w.r.t. the equivalence $c_{\mathcal{M}} \sim c_{\mathcal{N}}$ for all $c \in C$.

Categorically, the composition is a pushout:

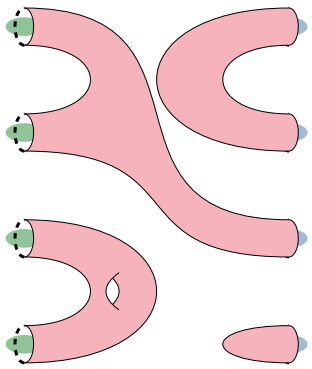


It is important to note that we only consider cobordisms here up to diffeomorphism, contrary to the original motivations where they are considered up to $(n+1)$ cobordisms (this would lead to ∞-categorical consideration, as $\mathbf{Cob}_{[n]}$ is hypothesized to be the free (∞,n)-compact closed category)

Example B.2 The category $\mathbf{Cob}_{[1]}$ is a simple example.
Its objects are closed smooth manifolds of dimension 0, that is finite sets of points, i.e. finite sets.
The set $\mathbf{Cob}_{[1]}(A,B)$ is then a smooth manifold of dimension 1 whose boundary is the disjoint union $A \sqcup B$, that is a collection of disconnected segments with endpoints in $A \sqcup B$ and circles. Here is an example of 1-cobordism:



Example B.3 Since the only connected manifold without boundary of dimension 1 is the circle, the objects in $\mathbf{Cob}_{[2]}$ are disjoint unions of circles. Here is an example of a 2-cobordism:



Proposition B.4 (Folklore) *The category* $\mathbf{Cob}_{[n]}$ *is dagger-compact.*

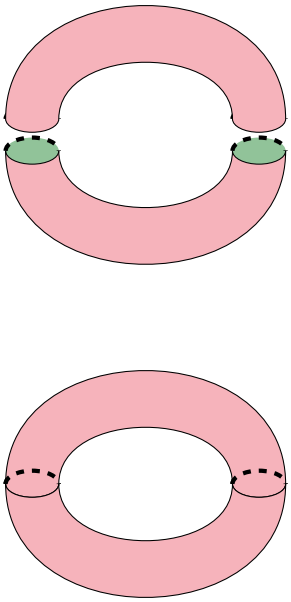
Corollary B.5 *In particular,* $\mathbf{Cob}_{[n]}$ *is compact closed (hence, also a traced monoidal category).*

About compact-closed category, there are two ways to make a model of LL out of them: by doing a double-glueing to get a "usual" semantic model, or do the **Int**-construction since they are traced, and then a double-glueing to get a GoI model. Is there a link between the two?

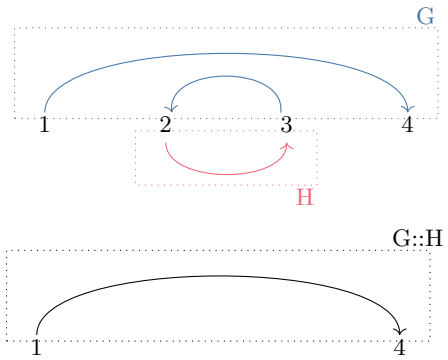
It seems that there is a bit of structure in common with **iG** (both categories are compact closed, also "geometrically" compose by a form of glueing). But as previously hinted, this category does not lose any scalar: the category $\mathbf{Cob}_{[1]}$ has a rich set of scalars: the manifolds without boundaries. For example, spheres and tori are morphism $\emptyset \to \emptyset$ in $\mathbf{Cob}_{[2]}$.

What do cobordisms have that interaction graph do not that they do not lose any scalars? The answer is hinted in the previous remark: manifolds *without boundaries*.

Example B.6 Glueing 2 C-shaped cobordisms give a donut:



Indeed: the problem with interaction graph is that *execution cares only about I/O*, and thus is lossy. Consider the following example which is an instance of 3.17:



The internal closed path between G and H is lost during execution because it *only cares about paths from the boundary of the graph to itself.*

Wagers are a way of reintroducing the information that was lost because the execution of interaction graph did not respect the geometry of the programs.

But there might be another way, which is the one that cobordism uses, that respects the geometry: *cospans, and glueing.* If the model is allowed to have programs without boundaries, then there is no need for wager or trefoil property anymore since no information will be lost.

Remark B.7 In dimension 1, which is the dimension of interaction graphs, the only manifold without boundary is the circle. Thus, in scalars, the only interesting thing that can happen topoligcally is to have

circles, and thus, counting circles (which is what the wager does) suffices.
If one finds an higher dimensional definition generalizing interaction graph, the monoid of wager would probably be more complex, and have elements such as 2 sphere + 3 tori.

If one looks again at how the category **ioGrph** is defined, the computation of the composition $G; H$ is actually done in 3 steps:

- First glue G and H together (this is a pushout, which is the way cospans are composed). This is very visible in the proof that the category is traced, where we consider the geometrical object and look at paths on it in most reasonings. This step, we call *the Glueing*. Note there is already some hiding at play here, in the sense that when composing $f : I \to C$ and $g : C \to O$ the internal part C becomes unnacessible to the outside thanks to the typing of category theory.
- Then we compute the paths (of length 2 or more). This step we call *the Saturation/Computing*. Together with the previous step, they are known in game semantics under the name *Composition*.
- Finally, we remove the internal workings, leaving only the paths. This step is called *the Hiding*.

Remark B.8 Two things are notable:

- The fact that cobordism are continuous makes it so that the Hiding is done automatically, since boundaries "merge" with the geometrical object once they are glued.
- The fact that they have elements without boundaries makes it so that fully internal closed paths are not forgotten.

Remark B.9 Something weird is at play in **iG**: the computational part, that is, the computing of paths, is duplicated during the definition: It happens both in the composition of **ioGrph** and in its trace, while one could expect to have a separation between the glueing and the computing.
Another thing is that the real "dynamics" are actually the computation of paths, while the glueing is more of a locative nature, but they are somehow mixed together.

If we want to make a model of computation with dynamics, we thus need a way to study how paths compose. It seems cobordisms and interaction graphs are glued in a similar way, but there is no computation of paths in cobordisms. We try to introduce paths in cobordisms in an ad-hoc way and see if we can take anything from it.

The Π_1 functor, which associates to a topological space X its fundamental groupoid, that is, the groupoid of all paths inside the space X, preserves certain pushouts of groupoids:

Theorem B.10 (Van Kampen's Theorem for Groupoid, Ronald Brown) *A set A is called representative in X if A meets each path-component of X. When $C = L \cap R$, and $\mathring{L} \cup \mathring{R} = X$, then:*

$$\begin{array}{ccc} \Pi_1(C, A) & \longrightarrow & \Pi_1(R, A) \\ \downarrow & & \downarrow \\ \Pi_1(L, A) & \longrightarrow & \Pi_1(X, A) \end{array}$$

is a pushout of groupoids.
In our setting, we will always take $A = X$ which is representative.
The proof of this version of the theorem can be found in[6].

We will now prove that because of this, one can "compute paths" in a functorial way.
For that we will use a central theorem to the theory of cobordisms:

Theorem B.11 (Collar Neighbourhood Theorem, Morton Brown) *Let X be a smooth manifold with boundary ∂X. Then the inclusion $\partial X \xrightarrow{i} X$ has an open neighbourhood $U \simeq_{diff} \partial X \times [0, 1)$ (an open collar).*
This fundamental theorem in the theory of cobordism was proven by Morton Brown in [5]

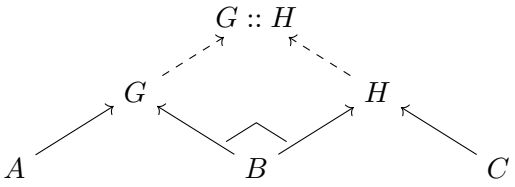
Definition B.12 We define the semi-category **ioGpd**:

Objects: Any set

Morphisms: Morphisms in $\mathbf{\imath o Gpd}(A, B)$ are groupoids over $A \sqcup B$ (up to isomorphism of groupoids, *not just equivalence*).

Composition: The composition of $G \in \mathbf{\imath o Gpd}(A, B)$ and $H \in \mathbf{\imath o Gpd}(B, C)$ is the (free) groupoid of paths from $A \sqcup C$ to $A \sqcup C$ in the glueing. We write it $G :: H$.

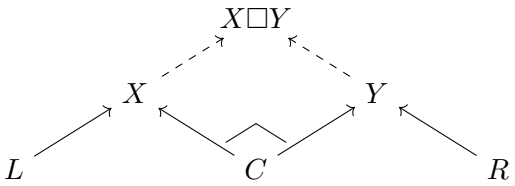
This can be presented as a pushout of cospans:



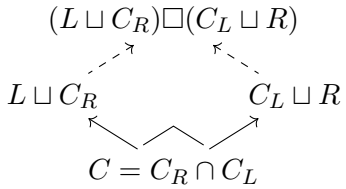
Note there is no identity, this is a *semi-category*.

Theorem B.13 *Π_1 is a semi-functor from $\mathbf{Cob}_{[n]} \to \imath o\mathbf{Gpd}$.*

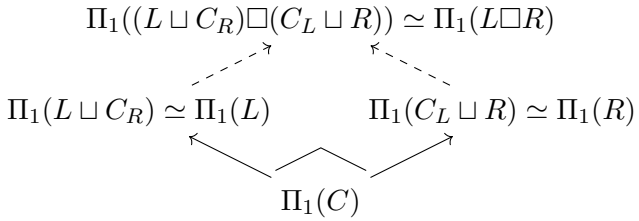
Proof. Take two cospans in $\mathbf{Cob}_{[n]}$ that can be composed together:



We then use the collar neighbourhood theorem on $(L \sqcup C \to X)$. This gives an open collar $O \simeq L \sqcup C \times [0, 1)$, from which we extract an open set $C_L \simeq C \times [0, 1)$. And similarly for $(R \sqcup C \to Y)$ to get a C_R.
We can form the following cospan:



Where we can apply Van Kampen's theorem:



All $\simeq$ are isos of groupoids being because glueing collars is just an equality when considering things up to diffeomorphism. (This is all thanks to the collar theorem, which allows to properly glue cobordisms together)
This diagram shows $\Pi_1(L \square R)$ is a pushout and thus we get our desired result: $\Pi_1(L \square R) \simeq \Pi_1(L) :: \Pi_1(R)$. □

Remark B.14 It seems that the Van-Kampen theorem is a proof that Π_1 is somehow analogous to the execution formula. This seems especially true when looking at 1-dimensional cobordism which are similar to a weak form of undirected interaction graph.

This suggests using a category similar to $\mathbf{\imath o Gpd}$ as a category to keep both the geometry and dynamics.

There is one thing of note here, the fact that cobordisms are not directed spaces, and thus the paths on cobordisms are invertible (hence the groupoid).
But this is not the case for graphs, which are more like directed spaces. The notion corresponding to fundamental groupoid in the directed case is the one of *fundamental category*. This suggests studying a category whose morphisms are not just graphs but semi-categories.

B.2 The curious case of Involutions

One might wonder what exactly is the status of $\mathbf{Cob}_{[n]}$, where does it fall in the main hierarchy of the paper that we presented?

Indeed, it is compact closed, like **iG**, but unlike **ioGrph**. But looking at $\mathbf{Cob}_{[1]}$, it seems that it is really similar to the category of Input/Output partial involutions (products of swaps) **ioInv** that we define here:

Definition B.15 We define the category **ioInv**

Objects: Any set.

Morphisms: $G : A \to B$ is a partial involution σ (think a product of disjoint swaps) on $A \sqcup B$ represented as an undirected graph. There is an edge $a - b$ if and only if $\sigma(a) = b$.

Composition: The composition of $G \in \mathbf{ioInv}(A, B)$ and $H \in \mathbf{ioInv}(B, C)$ is given by glueing the graph G and H along B.

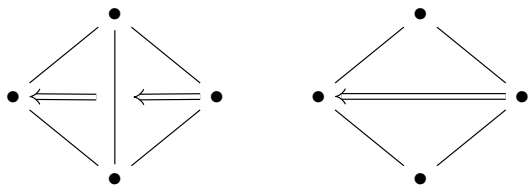
This category can easily be seen as a form of generalization of $\mathbf{Cob}_{[1]}$ that allows to have isolated vertices. This category **ioInv** allows to interpret programs since they are only made of axioms/cuts.

The advantage of **ioInv** is that it does not require directed homotopy, just regular homotopy. We could thus identify these morphisms up to homotopy equivalent (undirected) geometric realization.

Remark B.16 Notice how we did not require any form of obip representation. This makes the status of **ioInv** a bit unclear and this is discussed in the conclusion of this section.

ioInv is somehow weak in it's expressive power though, as it does not allow to interpret the correctness criterion. This would give a "weaker" model but it would still be interesting nonetheless.

Remark B.17 Since it seems models are about glueing along boundaries (here vertices): the model defined previously are about glueing directed/undirected graphs together, which is the 1-dimensional case of simplicial complexes. Thus there might be ways to generalize interaction graphs with things akin to simplicial sets. A naive idea would be to consider for example, two triangles, that could be glued along an edge to form a square with a diagonal, and the execution could remove said diagonal to form a square as in this example:



It might be possible to make a more general version of the model where we glue higher dimensional complexes on lesser dimensional boundaries, as in cobordism. As of today, the existence of a "right way" to make the dynamics of it work is an open problem.

Interaction Graphs generalise the models of Involution **ioInv**. As such one might wonder how does the simpler model fit in this setting?

We introduce for the discussion one last intermediate category between **ioInv** and **ioGrph** by allowing more general permutations than products of disjoint swaps:

Definition B.18 We define the category **ioPerm** as follows:

Objects: Any set.

Morphisms: $G : A \to B$ is a partial permutation σ on $A \sqcup B$ represented as an obip graph. There is an edge $a \to b$ if and only if $\sigma(a) = b$.

Composition: The composition of $G \in \mathbf{ioInv}(A, B)$ and $H \in \mathbf{ioInv}(B, C)$ is given by glueing the graph G and H along B and computing paths of lenght 2

Notice this also uses directed graphs.

Now in **ioInv**, contrary to **ioGrph** and **ioPerm**, but similarly to the category of 1-cobordisms $\mathbf{Cob}_{[1]}$, there is no use of an obip representation, the vertices of the graphs are *not* duplicated as X_s and X_t in this case, so it seems the constructions, while very similar, are of a different nature? In fact, the graph that one draws as a morphism is *directly* the graph that one wants to represent in **ioInv**.

There are many points of dicussions one could make, and it is very unclear what exactly is the relations between these two "worlds".

If seen from a really categorical perspective, **ioInv** and $\mathbf{Cob}_{[1]}$ are already compact closed, so one can already make a model of LL out of them by double glueing. At the same time, they seem to be superficially similar to **ioQuiv** and thus we would expect to do an **Int**-construction first before double glueing. This is possible because a compact closed category is traced. But then what would the resulting category be? What is the difference between $\mathbf{C}$ and it's free compact closure $\mathbf{Int}(\mathbf{C})$ when $\mathbf{C}$ is already compact closed? Will they give similar models of LL? If not, then one could iterate to get infinitely many different models... Would these be interesting.

From a more naive and computational perspective, we cannot really try and fit one into the other's perspective:

- The category **ioGrph** has to be defined in this way, for if it was defined in a way similar to **ioInv** there would be no way to enforce alternation.
- Reciprocally, it is unclear how one would construct a category similar to **ioGrph** for involutions, as there does not seem to be a natural obip representation in the undirected case. One could try and do a similar construction by duplicating vertices, and encoding a permutation (12) as an undirected edge $1_s - 2_t$ and $2_s - 1_t$, but this would duplicate everything and lose the information than these two edges were the same. From the program perspective, it is similar to compiling a swap to a graph $1 \leftrightarrow 2$, but an information is lost here: the fact that these two arrows are inverse of each other. One might think the setting of **ioCat** is suitable to retain such information since there can be equations between morphisms such as $f; g = id$, but it is not the case for the morphisms $1 \to 2$ and $2 \to 1$ actually have uncompatible types ($2_t \neq 2_s$).

The problem remains open, and ask a very strange question: how is it that the simpler case is actually very different to model using the traditional categorical tools of GoI? This is probably related to the two problems of locativity 3.8 section 3 and to the difference between the model of computation of Flows (which is directed) and of Stars (which is undirected) in Transcendantal Syntax ([23]).

Taking a final step back, we conclude this appendix with the following thoguht: it seems that even though Interaction Graphs are presented as a generalization of involutions, it is not completely the case:

Involutions are unable to express direction yes, but interaction graphs are *unable to express non-direction* (they can express having two arrows, but not the fact that they are inverse). Now, if one forgets about fitting the setting into category theory, **ioCat** still gives an interesting lesson, that one could probably replace Interaction Graphs by something along the lines of "Interaction Categories", with edges now composable and paths obtained as composition of edges. This setting would probably be more powerful and contain both models of Interaction Graphs (directed arrows), and Permutation (two arrows that are inverse of each other). One would know what the composition of $1 \to 2$ and $2 \to 1$ is, the identity at 1, but it would not be a "legal" composition somehow. The missing ingredient is to find out how to force alternation, as the previous paragraph shows the int-construction would be problematic (maybe colors?). This poses the qeustion as to wether alternation is even needed? Can one define a model generalising Interaction Graphs without the alternation requirement? It is indeed not necessary in the sister field of Game Semantics [7], where alternation is a specific restrictions ont he ay programs behave, but wild programs do not need to have alternation. And if this can be done, it is completely unclear if such a category can be generated from a space since it could be an arbitrary category.

C Omitted proofs

Proof. [Proof of Theorem 3.13] Since the category is symmetric monoidal, we have fewer axioms to verify. The proof is really easy to check visually but we still do it in details to show exactly what properties are used where. We use the notation e_{AB} means an edge is from A to B.

Consider $f : I \otimes X \to O \otimes X$, we will have to prove that two traced graphs are equals. We will use regular expressions to compare the paths $I \to O$ in the graphs before they are traced/composed (we will designate by "the shape of the graph" the "geometric" graph obtained by doing all necessary glueing but not computing the paths). Having the same paths, the graphs, once traced, will have same edges and thus be equal.

- **Naturality in I:** Given $g : L \to I$, we need to prove $\mathrm{Tr}^X_{I,O}(G)(g \otimes 1_X; f) = g; \mathrm{Tr}^X_{I,O}(G)(f)$ (see fig. 4a). Paths in the shape of $\mathrm{Tr}^X_{I,O}(G)(g \otimes 1_X; f)$ are of the form $e_g; e_{IO} + e_g; e_{IX}; (e_1 e_{XX})^*; e_1; e_{XO}$, while paths in the shape of $g; \mathrm{Tr}^X_{I,O}(G)(f)$ are of the form $e_g; (e_{IO} + e_{IX}; (e_{XX})^*; e_{XO})$. Clearly the paths of the shape described by these regular expressions are in one to one correspondence.
- **Naturality in O:** this case is symmetrical to the previous one.
- **Dinaturality in X:** Given $g : X \to X$, we need to prove $\mathrm{Tr}^X_{I,O}(G)(f; 1_O \otimes g) = \mathrm{Tr}^X_{I,O}(G)(1_I \otimes g; f)$. Before any reduction, paths in the shape of $\mathrm{Tr}^X_{I,O}(G)(f; 1_O \otimes g)$ are of the form $e_{IO}; e_1 + e^f_{IX}; (e^g_{XC}; e^f_{CX})^*; e^g_{XC}; e^f_{CO}; e_1$, while paths in the shape of $\mathrm{Tr}^X_{I,O}(G)(1_1 \otimes g; f)$ are of the form $e_1; e_{IO} + e_1; e^f_{IX}; (e^g_{XC}; e^f_{CX})^*; e^g_{XC}; e^f_{CO}$. There is a clear bijection between these paths, which shows that the two resulting graphs after tracing are the same.
- **Yanking:** We need to prove $\mathrm{Tr}^X_{X,X}(\sigma) = 1_X$. The paths inside $\square_X(\sigma)$ are of length 2 and are precisely the $e_{ij}e_{ji}$ with $i \in I, j \in X$, and j being the correspondent of i in the bijection $I \simeq X$. This means for each i there is a unique j giving such a path, and thus the resulting graph has one arrow from $i \to i$ for every i, thus it is 1.
- **Vanishing:** There are two case to consider. On one hand, vanishing for $\mathrm{Tr}^1_{I,O}(f) = f$ is obvious since $1 \otimes I = I$. On the other hand, $\mathrm{Tr}^{X \otimes Y}_{A,B}(f) = \mathrm{Tr}^X_{I,O}(\mathrm{Tr}^Y_{X \otimes I, X \otimes O}(f))$ is the most interesting case. The paths in the shape of $\mathrm{Tr}^{X \otimes Y}_{I,O}(f)$ can be "compacted", rearranging the parenthesis to form maximal subpaths only using edges e_{YY}. These subpaths are edges in $\mathrm{Tr}^Y_{X \otimes I, X \otimes O}(f)$, which makes this rearranging of parenthesis a path in $\mathrm{Tr}^X_{I,O}(\mathrm{Tr}^Y_{X \otimes I, X \otimes O}(f))$.
- **Superposing:** We need to prove $\mathrm{Tr}^X_{C \otimes A, C \otimes B}(1_C \otimes f) = 1_C \otimes \mathrm{Tr}^X_{A,B}(f)$ (see fig. 4b). By a case disjunction, it is clear that either a path falls in 1_C and it is trivial or it falls in f and it is as usual.

□