Drifter;Graphs : a dynamically-sliced model for Additives

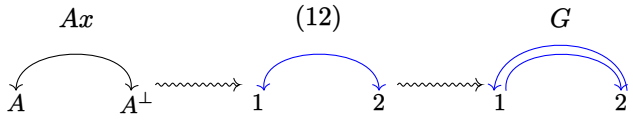
Valentin Maestracci

May 28, 2025

Linear realisability [6] is a research program in the continuation of Geometry of Interaction, where logic is studied from the point of view of models of computation. Many such models of computation have been defined (ludics, flows, stellar resolution...), in particular the family of interaction graphs. We define a new type of interaction graph: drifter graphs, that is designed to handle additives in a finer way than the regular construction. We do so in a new framework, using diagrams, that allows to unify all models of these family as similar constructions.

1. Interaction Graphs

Interaction Graphs are a model of geometry of interaction introduced by Seiller in [5]. They generalize another model, the *model of permutations* which encodes proofs nets of Multiplicative Linear Logic as permutations [1], and the dynamic of cut elimination by making permutation acts on atoms and following where they end up. Interaction Graphs generalize this by considering arbitrary graphs:



Notation 1 *We note $\ominus$ the symmetric difference of sets.*

We assume given a set $\mathtt{C}$ of colors.

Definition 1 (Colored (Directed) Graph) *A colored directed graph is a directed graph (V, E, s, t) equipped with a painting function $p : E \to \mathtt{C}$. We say that a graph is n-colored when $|C| = n$.*

Definition 2 (IG) *An interaction graphs is a 1-colored directed graph. We say an interaction graphs G it is* at (location) *L when $V_G = L$. We will write $(IG)_L$ to designate the set of interaction graphs at L and more generally S_L for any set $S \subseteq (IG)_L$ of interaction graphs at L.*

This definition is not the standard definition of interaction graphs, but it is easily verified to be equivalent. We will define in here a simple generalization called bicig (standing for bicolored interaction graphs). Beforehand, we define how graphs interact, as to define a model of computation.

2. Diagrams

We will try to compute paths in the graph. Those that are of interest to us will be the alternating paths, which corresponds to an alternation between axioms and cuts. This is the reason we introduced colors: an alternating path will be a path in which the color changes at every edge. We introduce the notion of diagram, which will correspond to said path. This notion might seem a bit

of an overkill for this purpose, but the abstract definitions we introduce here can be used to define other model of computation (flows, stellar resolution, etc.), where they are necessary, and providing a uniform "framework".

Diagrams will represent an *attempt at a composition* of our elementary objects, here edges. (Which might fail, if the source and targets do not match.)

Definition 3 (Diagram) *A diagram δ on a directed graph G is the data of a graph H, called* the shape *of δ, whose connected components are linear, that is of the shape $- \to \ldots \to -$ and with at least one edge. And a function $\delta : E_H \to E_G$, selecting an edge of G to be "above" every edge of H. (Note how we conflate the name of the diagram and the name of said data).*
We will thus write such shapes as $- \xrightarrow{e} -$ to indicate that $\delta(e_H) = e \in G$.
Note how linearity implies that between two vertex v, w there is at most one arrow. We will write $v \to w$ to designate this arrow (and implicitly assuming it exists). We will sometimes write $v \to -$ when we do not want to name of the vertices.

Remark 1 *In a non connected diagram, the fact that there are multiple connected components could be interpreted as doing "attempts" at computing paths in parallel.*

Definition 4 (Boundary of a Diagram) *Given a diagram D, its source boundary is the set $(\partial\delta)_s := \{v \in V_\delta \mid \text{there is no } e \text{ with } t(e) = v\}$. Similarly, its target boundary is the set $(\partial\delta)_t$ defined as $\{v \in V_\delta \mid \text{there is no } e \text{ with } s(e) = v\}$. Its boundary $\partial\delta$ is the union $s(\partial\delta)_s \sqcup (\partial\delta)_t$ (disjoint since there is no lonely vertex). Its located boundary $(\partial\delta)_{loc}$ is defined as $\{s(\delta(v \to -))\} \cup \{t(\delta(- \to v))\}$.*

Definition 5 (Interior of a Diagram) *Given a diagram δ, we define its interior δ° as the set $\{v \in V_\delta \mid - \to v \to -\}$. We define its located interior as $(\delta^\circ)_{loc} := \left\{ t(\delta(e)), s(\delta(e')) \;\middle|\; - \xrightarrow{e} v \xrightarrow{e'} - \right\}$.*

Definition 6 (Equation) *Given a vertex v in a diagram such that $\ldots \xrightarrow{e} v \xrightarrow{e'} \ldots$, define the* equation at v *as the equation $eq(v) := t(e) \stackrel{?}{=} s(e')$. This tests wether the arrows are compatible or not. This can be extended to any set V immediately to get an unification problem: $eq(V) := \{eq(v) \mid v \in V\}$.*

Definition 7 (Locativity: On Location and Saturation) *Given L a set of locations, a diagram δ is said to be* on L *when $(\delta^\circ)_{loc} \subseteq L$. A diagram is L-saturated when $(\partial\delta)_{loc} \cap L = \emptyset$. We write $\texttt{Diags}_L(G)$ the set of diagrams of G that are on L and $\bar{L}$-saturated.*

Note 1 *Notice that saturation formalises the fact that one cannot compose by an edge anymore.*

Definition 8 (Compatible) *A diagram δ is said to be compatible when $eq(\delta^\circ)$, seen as a unification problem, has a solution.*

Note 2 *Notice that in the context of graphs we are considering, compatibility only means that all equations are equalities, as unification is not really used.*

Definition 9 (Alternated Diagram) *A diagram δ for a graph G is said to be alternated when for all e, f in configuration $- \xrightarrow{e} - \xrightarrow{f} -$, $p(e) \neq p(f)$ in G.*

Definition 10 (Correction) *A diagram δ is said to be correct when it is* compatible, connected, *and* alternated. *We write $\texttt{CDiags}_L(G)$ for the subset of $\texttt{Diags}_L(G)$ of correct diagrams of G. We write $\Downarrow \delta$ (the reduction of delta) for the edge $e_0; \ldots; e_n$ (here it is just a formal path, but in later consideration there will be a composition of arrows).*

Proposition 1 (Overkill) *A correct diagram, in the case of interaction graphs, is exactly an alternated path.*

The point of this notion was to introduce the notion of *saturated diagrams*: these describes paths inside the cuts that have endpoints outside of it.

3. How to compose programs / graphs, and dynamics

Execution between two graphs amounts to computing the alternating paths in a graph obtained by glueing the two graphs together.

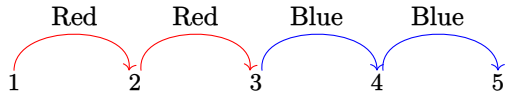
Definition 11 (Glueing) *Given two graphs F, G of disjoint colors, one can define the colored directed graph $G \between F$ as their (*non necessarily disjoint*!) locative union, that is:*

$$G \between F := (V_G \cup V_F, E_G \sqcup E_F, s_G \sqcup s_F, t_G \sqcup t_F) \tag{1}$$

The coloring is given by $p = p_F \sqcup p_G$ (this is why we need disjoint colors). Notice this is not an interaction graphs *(it has two colors). More generally, we can define a glueing of graphs for an arbitrary (albeit with different colors) set of graph.*

The main result we want to prove is associativity of execution (confluence): $\mathrm{Ex}(\mathrm{Ex}(A,B),C) = \mathrm{Ex}(A,B,C) = \mathrm{Ex}(A,\mathrm{Ex}(B,C))$, which also involves the glueing of three graphs (which has three colors). It is thus convenient to directly consider colored graphs as the computational notion in itself.

There is a problem considering simple colored graphs if we want to be able to do "progressive reductions", like a small step semantics. Imagine composing the two edges in the following graph:



What color should the resulting arrow have? Adding a fresh color would not help, because we need to remember that we cannot compose with red on one side, and blue on the other. We thus need to consider edges that are bicolored. The philosophy that we apply to our construction is that somehow there should be no difference between an edge (reduced diagram) and a diagram.

Definition 12 (Bicolored Interaction Graphs (bicig)) *A bicig is a directed graph equipped with a painting function $p : E \times \{s,t\} \to \mathbf{C}$.*

Notice that mono-colored bicig correspond exactly to interaction graphs. We are thus going to generalize the previously defined constructions to the bi-colored case.

Definition 13 (Alternating Paths in bicig) *A path in a bicig is said to be alternated when $p(e_i)(t) \neq p(e_{i+1})(s)$.*

Definition 14 (Execution of a bicig) *Given a bicig G and a set of locations L, define $\mathrm{Ex}_L(G)$ by letting:*

$$V_{\mathrm{Ex}_L(G)} = L_G \cap \bar{L}, \qquad E_{\mathrm{Ex}_L(G)} = \Downarrow \mathtt{CDiags}_L(G),$$

with obvious s and t maps, and coloring function $p_{\mathrm{Ex}_L(G)} : e, x \in E \times \{s,t\} \to p_G(x(e))(x)$.

Logically, this amounts to consider L as the set of locations within cut formulas and performing cut-elimination.

Note 3 *Note for interaction graphs, coloring G with color G and H with color H, we have $\mathrm{Ex}(G,H) = \mathrm{Ex}_{L_G \cap L_H}(G \between H)$ (forgetting the colors on the right). This show that this a first step toward a sort of microscopic decomposition of the execution formula, where one could compute cuts independently*

Proposition 2 (Church-Rosser) *When $L \cap K = \emptyset$, we have the Church-Rosser property:*

$$\mathrm{Ex}_L(\mathrm{Ex}_K(H)) = \mathrm{Ex}_{L \sqcup K}(H) = \mathrm{Ex}_K(\mathrm{Ex}_L(H)).$$

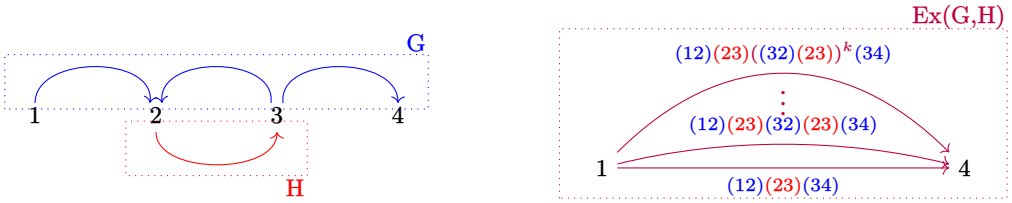


Figure 1: Two graphs and their execution

Proposition 3 ((Classical) Associativity of Execution) *When $L_A \cap L_B \cap L_C = \emptyset$, we have:*

$$\mathrm{Ex}(\mathrm{Ex}(A,B),C) = \mathrm{Ex}(A,B,C) = \mathrm{Ex}(A,\mathrm{Ex}(B,C)).$$

With explicit locations[1]*:* $\mathrm{Ex}_{\overline{L_A \ominus L_B \ominus L_C}}(A,B,C) = \mathrm{Ex}_{(L_A \ominus L_B)\cap L_C}(\mathrm{Ex}_{L_A \cap L_B}(A,B),C)$.

Remark 2 (Comparison of this approach with the traditional one) *This approach with diagrams is of a more "dynamically located" nature: we attempt to compose edges at runtime to compute a path, but composition might fail. The traditional approach is more "statically located": the edges are already drawn on the paper at the right locations at "compile time", and we just compute the paths inside the geometric graph. There is a way to get the "best of both worlds" through a notion called "dependency graph".*

4. Drifter Graphs: A new model of additivity and thickness

Traditionally, additives are handled via a slice construction. We follow a construction similar in spirit but different in its execution that solves some (although not all) of the problems of the usual slice constructions. We will directly perform the construction corresponding to "thick graphs" [4] here, that is additive but with arrows that can go from a slice to another. The purely additive construction can be deduced from this one.

Definition 15 (World Lines) *In this section, we are given a set $\mathscr{W}_l$ called the set of world lines (this is similar to the set of indices for the additive slices). A worldline is an element of $\mathscr{W}_l$.*

Definition 16 (Tracker) *A worldline-tracker, abbreviated as tracker is a partial function $\mathbf{C} \rightharpoonup \mathscr{W}_l$. Those are used to keep track of which worldline we are exploring in each graph. (Note how a color is unique to a graph, so it is used as an "identity card" for it).*

We now define some operations on trackers:

Definition 17 (Coherence of trackers) *We say that two trackers T, T' are coherent on S a set, written $T \frown_S T'$, when $\forall u \in S, T(u) = T'(u)$* whenever they are both defined. *In particular we write $T \frown T'$ to say $T \frown_{\mathrm{dom}(T)\cap\mathrm{dom}(T')} T'$.*

We now introduce the model of computation that we are interested in.

Definition 18 (A drifter bicig) *A* drifter bicig *is a bicig with the additional data of a function $T : E \times \{s,t\} \to (\mathbf{C} \to \mathscr{W}_l)$. It will be used to track worldlines along time (composition of arrows).*

Notation 2 (Guarding and Drifting trackers) *We will write G_i for trackers of $T(-,s)$, because they will be used as "guardians", to check wether it is consistent to take an arrow or not when following a path. (We must check that arrows were in the same "slice".) Their composition will be called* fortification. *We will write D_i for trackers of $T(-,t)$, because they will be used to "drift" from the current worldline (change of slice). Their composition will be called* overwritting. *Both fortification and overwriting are actually just "special cases" of an operation called overriding, defined below.*

[1] The other equality follows by symmetry.

Intuition 1 *When following a path inside the graph, we will keep track of the different worldlines that each programs are in, in a linetracker T_0. To accept a new edge e, we will require that $T_0 \frown G_0 := T(e,s)$, that is, in the current worldline, we can indeed use e. We will then update T_0 to a T_1, with the data of $D_0 := T(e,t)$, which will allow us to change worldlines.*

To do all that, we need to understand how to compose arrows:

Definition 19 (Compatible Arrows) *A pair of arrows:*

$$\bullet \xrightarrow{G_0|D_0} \bullet \xrightarrow{G_1|D_1} \bullet$$

is timeline coherent *(or* compatible*) when $D_0 \frown G_1$ and $G_0 \frown_{\overline{D_0}} G_1$. That is: D_0 passes the test of G_1, and the tests that are not overwritten by D_0 in G_0 are compatible with those in G_1; if not then one could not pass both.*

Definition 20 (Overriding) *Given trackers G_0, D_1, G_1, we define the partial overriding operation (defined when said trackers come from compatible arrows as in the previous definition)*

$$G_1 <<_{D_1} G_0 : u \to \begin{cases} G_1(u) & \text{if } u \in \overline{\mathrm{dom}(G_1)} \cap \overline{\mathrm{dom}(D_1)} \\ G_0(u) & \text{otherwise} \end{cases}$$

Definition 21 (Composition of Arrows) *When two arrows are compatible, their composition is defined as*

$$\bullet \xrightarrow{G_1 << D_0 \, G_0|D_0 << D_1} \bullet$$

Intuition 2 *To drift, one drifts along D_0 and then along D_1, which might override what D_0 did. To guard, one must pass first the test of G_0, then if a worldline is not overridden by D_0 we test it with G_1, explaining why the composition is in reverse.*

Proposition 4 (Associativity of Composition / Overriding) *Given trackers G_0, G_1, G_2 and D_0, D_1, we have:*

$$G_2 <<_{D_0 << D_1} (G_1 <<_{D_0} G_0) = (G_2 <<_{D_1} G_1) <<_{D_0} G_0$$

This might not look like an associativity from the overriding point of view, since there is an extra $D_0 << D_1$ term, but it actually is an associativity from the composition of arrow point of view.

We now have all the ingredients needed to define *timeline correctness.*

Definition 22 (Timeline Correctness) *A diagram $\delta = v_0 \overset{e_0}{\to} v_1 \overset{e_1}{\to} \cdots v_n$ is said to be* timeline correct *when the composition $e_0; \ldots; e_{n-1}$ is defined.*

Definition 23 (Correct Diagram) *A diagram for a drifter bicig is correct when it is correct in the sense of bicig and timeline correct.*

If a diagram is correct, then we can compose its arrows both from the point of view of usual locativity and of the timeline, we can thus define an actualisation:

Definition 24 (Actualisation) *Suppose given a diagram $\delta = v_0 \overset{e_0=G_0,T_0}{\to} v_1 \overset{e_1=G_1,T_1}{\to} \cdots v_{n+1}$ and let $e = G, T = e_0; \ldots; e_{n-1}$; that is, $T := T_0 << T_1 << \ldots$ and $G := G_n <<_{T_{n-1}} G_{n-1} \ldots$. Then the actualisation of δ is $\Downarrow \delta := v_0 \overset{e=G,T}{\to} v_{n+1}$.*

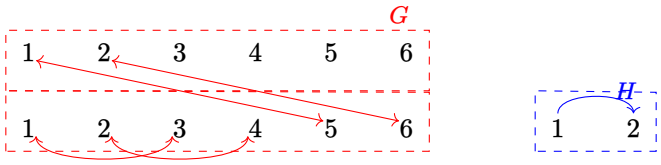
From these definitions we can reprove everything that was defined for bicig (associativity and lemmas) keeping the same definitions.

A. Appendix: What problem do Drifter Graphs Solve?

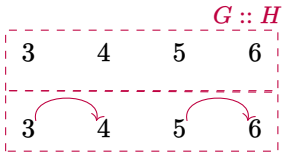
They do not solve the *"problem of additives"*: by that we mean the fact that when doing a cut of $\Gamma \multimap A \,\&\, B$ on a $A \oplus B \multimap \Delta$ coming from an A, the part internal to Γ in the slice of B appears as a form of "leftover" after execution. This is dealt with by Seiller by showing that these leftovers are "invisible" from the point of view of interaction, and thus can be quotiented out. [3] This was also dealt with by Hamano [2] which used a "global" approach to slices, which allowed to erase a slice that is not where interaction is happening.

But they do solve a problem that the additive construction of Seiller has: the one of "empty slices".

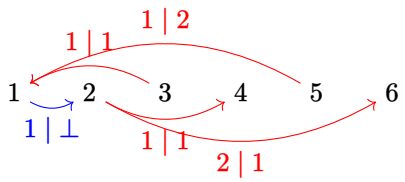
Consider the contraction program (on the left) against a simple program (right):



This normalizes to a graph with an empty slice:



But in the case of drifter graphs (we represent only the relevant edges here, and we do a monocolor consideration) this "slice" would disappear, as in:



(a) A blue graph against contraction

(b) The result of execution

And thus any mention of the worldline 2 has disappeared: this is because $1|2; 2|1$ gives $1|1$.

References

[1] Jean-Yves Girard. "Multiplicatives". In: *Logic and Computer Science: New Trends and Applications*. Ed. by G. Lolli. Rosenberg & Sellier, 1987, pp. 11–34.

[2] Masahiro Hamano. "A MALL Geometry of Interaction Based on Indexed Linear Logic". In: *Mathematical Structures in Computer Science* 30.10 (Nov. 2020), pp. 1025–1053. URL: https://www.cambridge.org/core/journals/mathematical-structures-in-computer-science/article/mall-geometry-of-interaction-based-on-indexed-linear-logic/94895D6D28B8CD45F314E2D21CD9C7B9 (visited on 05/20/2025).

[3] Thomas Seiller. "Interaction Graphs: Additives". In: *Annals of Pure and Applied Logic* 167.2 (Feb. 1, 2016), pp. 95–154. URL: https://www.sciencedirect.com/science/article/pii/S0168007215000998 (visited on 03/17/2025).

[4] Thomas Seiller. "Interaction Graphs: Exponentials". In: *Logical Methods in Computer Science* Volume 15, Issue 3 (Aug. 30, 2019). URL: https://lmcs.episciences.org/5730 (visited on 03/17/2025).

[5] Thomas Seiller. “Interaction Graphs: Multiplicatives”. In: *Annals of Pure and Applied Logic* 163.12 (Dec. 1, 2012), pp. 1808–1837. URL: https://www.sciencedirect.com/science/article/pii/S0168007212000759 (visited on 03/17/2025).

[6] Thomas Seiller. “Mathematical Informatics”. thesis. Université Sorbonne Paris Nord, June 18, 2024. URL: https://theses.hal.science/tel-04616661 (visited on 03/17/2025).