A Structural and Nominal Syntax for Diagrams

Dan R. Ghica
University of Birmingham

Aliaume Lopez
ENS Cachan, Université Paris-Saclay

The correspondence between monoidal categories and graphical languages of diagrams has been studied extensively, leading to applications in quantum computing and communication, systems theory, circuit design and more. From the categorical perspective, diagrams can be specified using (name-free) combinators which enjoy elegant equational properties. However, conventional notations for diagrammatic structures, such as hardware description languages (VHDL, VERILOG) or graph languages (DOT), use a different style, which is flat, relational, and reliant on extensive use of names (labels). Such languages are not known to enjoy nice syntactic equational properties. However, since they make it relatively easy to specify (and modify) arbitrary diagrammatic structures they are more popular than the combinator style. In this paper we show how the two approaches to diagram syntax can be reconciled and unified in a way that does not change the semantics and the existing equational theory. Additionally, we give sound and complete equational theories for the combined syntax.

1 Specifying graphs

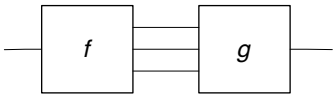
Graphs and their visual representations (diagrams) are an appealing way of describing many kinds of systems, in particular circuits. Work originated in the study of quantum computation [1] has exploited the connection between various classes of graphs and monoidal categories, going back to the seminal work of Joyal and Street [14], to add a layer of structure which makes reasoning about and with diagrams not just intuitive but also mathematically rigorous. Subsequently this connection was extended in many surprising and interesting directions, from computational linguistics [17], to modelling signal flow [4] and synchronous [9] or asynchronous [8] circuits. New automated reasoning tools based on diagrams, rather than the usual linear algebraic syntax, are a particularly exciting development (see `http://globular.science/`). This convenient and elegant interplay between the dual categorical and diagrammatic methods are by now a mature and rich area of research [18].

Although these developments convincingly establish the usefulness of categorical diagrammatics in reasoning about many kinds of systems, we note that little has been suggested in terms of a workable syntax which is conceptually compatible with it, but also with conventional notations, which are unstructured and relational. Examples of the latter are hardware description languages such as VERILOG and VHDL, or graph languages such as DOT (see `http://www.graphviz.org/`). Indeed, in the literature the categorical combinators are usually taken as an implicit diagram syntax. Whereas such a syntax is expressive enough to describe the desired classes of graphs, it is often not as convenient as the alternatives. Although categorical combinators can be elegant and succinct in certain situation, *point-free* languages of combinators generally hold little appeal as concrete syntax (e.g. APL).

Conventional diagram syntax lacks structure. It is a "flat" relational description of the graph. Whereas the categorical combinators are name-free, conventional syntax uses an abundance of names, for nodes and sometimes even for edges. Although these languages are widely used, they are broadly considered both inelegant and unwieldy, and purely structural alternatives have been proposed [2].

Bob Coecke and Aleks Kissinger (Eds.):
14th International Conference on Quantum Physics and Logic (QPL)
EPTCS 266, 2018, pp. 71–83, doi:10.4204/EPTCS.266.4

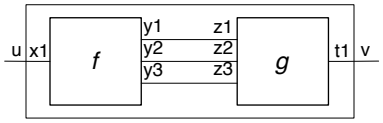
Before diving into technicalities let us consider a simple motivating example. Consider the diagram below, consisting of two components f and g, the first with one input and three outputs and second with three inputs and one output, connected in the obvious way:



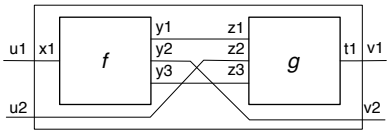
In the categorical, combinator-style, specification such a diagram would be succinctly written as the *composition* $f;g : 1 \to 1$, of $f : 1 \to 3$ and $g : 3 \to 1$. A VHDL-style description would look more verbose:

```
module fg(input u; output v)
begin
  component f(input x1; output y1, y2, y3);
  component g(input z1, z2, z3; output t1);
  wire y1, z1; wire y2, z2; wire y3, z3; wire u,  x1; wire t1, v;
end
```

This is the diagram annotated with all the names used above:



In highly structured diagrams the categorical notation is concise and elegant. However, realistic circuits may also use arbitrary connections. Consider a variation of the circuit above:



The flat description can be adjusted to cope with arbitrary connector reassignments:

```
module fg(input u1, u2; output v1, v2)
begin
  component f(input x1; output y1, y2, y3);
  component g(input z1, z2, z3; output t1);
  wire y1, z1; wire y3, z3; wire y2, v2; wire u2, z2; wire u1, x1; wire t1, v1;
end
```

The categorical style description is now more intricate, requiring a mix of sequential (;) and parallel (*tensor*, $\otimes$) composition along with the use of *structural* combinators such as the single connector (*identity*, i) and crossing connector (*symmetry*, γ): $(f \otimes i);(i \otimes \gamma \otimes i);(i \otimes i \otimes \gamma);(i \otimes \gamma \otimes i);(g \otimes i)$.

The combinator-style variable-free syntax is appealing for highly structured diagrams and awkward for arbitrary ones, whereas the flat and unstructured syntax seems useful in the case of unstructured and unnecessarily verbose for structured graphs. Without advocating one style or the other, in this paper we present a syntax which shows that the two are compatible. Our contributions are therefore as follows:

1. Preserving and extending the equational theory.
2. Possibly eliminating the need for structural combinators.
3. Preserving and conserving the expressiveness of the underlying diagrams.
4. Reasoning equationally in the new syntax.

$$\frac{}{x \mid - \vdash x : 0 \to 1 \mid x \leq x} \qquad \frac{}{- \mid x \vdash x : 1 \to 0 \mid x \leq x} \qquad \frac{}{- \mid - \vdash k : m \to n \mid \emptyset}$$

$$\frac{\Gamma \mid \Delta \vdash M : m_1 \to m_2 \mid \leq \qquad \Gamma' \mid \Delta' \vdash N : n_1 \to n_2 \mid \leq'}{\Gamma \uplus \Gamma' \mid \Delta \uplus \Delta' \vdash M;N : m_1 \to n_2 \mid [\leq \cup \leq' \cup ((\Gamma \uplus \Delta) \times (\Gamma' \uplus \Delta'))]}\ m_2 = n_1$$

$$\frac{\Gamma \mid \Delta \vdash M : m_1 \to m_2 \mid \leq \qquad \Gamma' \mid \Delta' \vdash N : n_1 \to n_2 \mid \leq'}{\Gamma \uplus \Gamma' \mid \Delta \uplus \Delta' \vdash M \otimes N : m_1 + n_1 \to m_2 + n_2 \mid [\leq \cup \leq']}$$

$$\frac{\Gamma \mid \Delta \vdash M : m_1 \to m_2 \mid \leq \qquad [\leq \cup \{(x,y)\}] \text{ is a partial order}}{\Gamma \mid \Delta \vdash M : m_1 \to m_2 \mid [\leq \cup \{(x,y)\}]}\ x, y \in \Gamma \uplus \Delta$$

$$\frac{\Gamma, x \mid \Delta, y \vdash M : m_1 \to m_2 \mid \leq \qquad x \leq y}{\Gamma \mid \Delta \vdash \overline{xy}.M : m_1 \to m_2 \mid \leq \setminus \{x,y\}}$$

Figure 1: Uniflow diagrams typing rules

2 Uniflow diagrams

A PROP (abbreviation of *products and permutations*) is a strict symmetric monoidal category where every object is a natural number [11]. We give a graph semantics of PROPs based on Kissinger's *framed point graphs* [15]. Let a labelled directed acyclic graph (DAG) be a DAG (V,E) equipped with a partial injection $f : V \rightharpoonup L$ and a relation $E \subseteq V^2$ such that the transitive and reflexive closure of E is a partial order on V. Let a labelled interfaced DAG (LIDAG) be a labelled DAG with two distinguished lists of unlabelled nodes representing the "input" and "output" ports. Unlabelled nodes are called *wire nodes*, and edges connecting them are called *wires*. A wire homeomorphism [15, Sec. 5.2.1] is any insertion or removal of wire nodes along wires which does not otherwise change the shape of the graph. Two LIDAGs are considered to be equivalent if they are graph isomorphic up to renaming vertices and wire homeomorphisms. The quotienting of LIDAGs by this equivalence gives us *framed point DAGs* [15, Def. 5.3.1], which we will call *uniflow diagrams*. We give a syntax for uniflow diagram as follows:

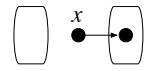
$$M ::= k \mid i \mid \gamma \mid M;M \mid M \otimes M \mid x \mid \overline{xy}.M,$$

where $k \in K$ are the constants and x variables. The language is essentially that of symmetrical monoidal categorical combinators (identity, symmetry, composition, tensor) over a signature, extended with variables and a binding construct we read as *link x and y in M*. We equip this language with a type system, where judgements are of the form $\Gamma \mid \Delta \vdash M : m \to n \mid R$. Γ is a set of *input variables*, Δ of *output variables* and R a partial order on their (disjoint) union, called the *anchor* of the diagram (Fig. 1). The anchor relation is a syntactic discipline used to prevent the inadvertent introduction of cycles. Note that $i : 1 \to 1$ and $\gamma : 2 \to 2$ have the same typing rules as the constants. The type $m \to n$ represents a uniflow diagram with m (unlabelled) inputs and n (unlabelled) outputs. Given a relation R we denote by $[R]$ its transitive and reflexive closure. For any set S, we define $R \setminus S = \{(x,y) \in R \mid x,y \notin S\}$.

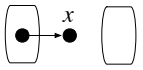
We give a concrete graph-theoretic semantics of the uniflow diagram. We may write $\Gamma \mid \Delta \vdash M$ if $\Gamma \mid \Delta \vdash M : m \to n \mid R$ for some $m,n \in \mathbb{N}$ and some partial order R. The semantics is defined by induction on the typing derivation. The meaning of a judgement $\Gamma \mid \Delta \vdash M : m \to n \mid R$ is a uniflow diagram $\left(V,I,O,E,f : V \rightharpoonup K \uplus \mathbb{A}\right)$ such that $\mathbb{A} \supseteq \Gamma,\Delta$ is a set of names and the relation R is a partial order on the vertices labelled by variables $\Gamma \uplus \Delta$ which is compatible with E, i.e. if $(x,y) \in R$ then there exist unique vertices u,v such that $f(u) = x, f(v) = y$ and *there is no path* in E from v back to u. We write this as $[\![\Gamma \mid \Delta \vdash M : m \to n \mid R]\!] = (V,I,O,E,f) \asymp R$. The structural combinators are interpreted as in Fig. 2.

Let :: stand for list concatenation or cons-ing. And let the "zipping" of two lists be defined in the

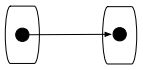
$$[\![x \mid - \vdash x : 0 \to 1 \mid x \leq x]\!] = (\{a,b\}, nil, b :: nil, \{(a,b)\}, \{(a,x)\}) \asymp \{(a,a)\} =$$



$$[\![- \mid x \vdash x : 1 \to 0 \mid x \leq x]\!] = (\{a,b\}, a :: nil, nil, \{(a,b)\}, \{(b,x)\}) \asymp \{(b,b)\} =$$



$$[\![- \mid - \vdash i : 1 \to 1 \mid \emptyset]\!] = (\{a,b\}, a :: nil, b :: nil, \{(a,b)\}, \emptyset) \asymp \emptyset =$$



$$[\![- \mid - \vdash \gamma : 2 \to 2 \mid \emptyset]\!] = (\{a,a',b,b'\}, a :: a' :: nil, b :: b' :: nil, \{(a,b'),(a',b)\}, \emptyset) \asymp \emptyset =$$

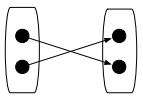


Figure 2: Interpreting structural combinators

usual way ($zip\ nil\ nil = nil$, and $zip\,(x :: xs)\,(y :: ys) = (x,y) :: (zip\ xs\ ys)$). If unambiguous we may use a list to also mean the set of elements of that list.

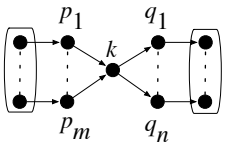
Assuming that $[\![\Gamma_i \mid \Delta_i \vdash M_i : m_i \to n_i \mid R_i]\!] = (V_i, I_i, O_i, E_i, f_i) \asymp R_i$, for $(i = 1,2)$, The interpretation of the two forms of composition is:

$$\begin{aligned}
&[\![\Gamma_1 \uplus \Gamma_2 \mid \Delta_1 \uplus \Delta_2 \vdash M;N : m_1 \to n_2 \mid [R_1 \cup R_2 \cup (\Gamma_1 \uplus \Delta_1) \times (\Gamma_2 \uplus \Delta_2)]]\!] \\
&\qquad = (V_1 \uplus V_2, I_1, O_2, E_1 \cup E_2 \cup (zip\ O_1\ I_2), f_1 \cup f_2) \asymp [R_1 \cup R_2 \cup (\Gamma_1 \uplus \Delta_1) \times (\Gamma_2 \uplus \Delta_2)] \\
&[\![\Gamma_1 \uplus \Gamma_2 \mid \Delta_1 \uplus \Delta_2 \vdash M \otimes N : m_1 + m_2 \to n_1 + n_2 \mid [R_1 \cup R_2]]\!] \\
&\qquad = (V_1 \uplus V_2, I_1 :: I_2, O_1 :: O_2, E_1 \cup E_2, f_1 \cup f_2) \asymp [R_1 \cup R_2].
\end{aligned}$$

Note that since diagrams are defined up to relabelling nodes, these definitions are always good. In case of name clashes nodes can be given fresh names.

Constants are interpreted as below, with p_i, q_j labels for the input and output ports of the constant.

$$[\![- \mid - \vdash k : m \to n \mid \emptyset]\!] =$$



The rule for strengthening the anchor is interpreted as

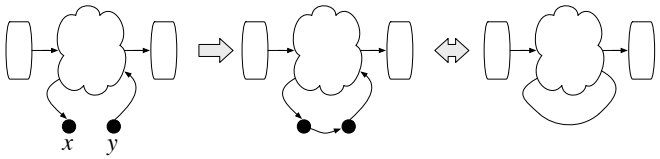
$$[\![\Gamma \mid \Delta \vdash M : m \to n \mid [R \cup \{(x,y)\}]]\!] = (V, I, O, E, f) \asymp [R \cup (f^{-1}(x), f^{-1}(y))].$$

The restriction on $[R \cup (f^{-1}(x), f^{-1}(y))]$ being a proper partial order ensures that the interpretation of the rule is well defined.

Let $(f \setminus B')(x) = f(x)$ if $f(x) \notin B'$ and undefined otherwise. The new construct is the *link*:

$$[\![\Gamma \mid \Delta \vdash \overline{xy}.M : m \to n \mid R \setminus \{x,y\}]\!] = (V, I, O, E \cup (f^{-1}(x), f^{-1}(y)), f \setminus \{x,y\}) \asymp R \setminus \{(x,y)\}. \quad (1)$$

The link can be visualised as below, including the wire homeomorphism:



Lemma 1 (Soundness)**.** *Any well typed term denotes a valid uniflow diagram.*

The useful connection between diagrams and monoidal categories is given by this proposition:

Proposition 2. *Given a term $\Gamma \mid \Delta \vdash M : m \to n$, the diagram $[\![\Gamma \mid \Delta \vdash M : m \to n]\!]$ is a morphism in the strict free symmetric monoidal category (SMC) over signature $K \cup \Gamma \cup \Delta$ of constants and (uninterpreted) variables.*

By "uninterpreted variables" we mean simply that the variables from Γ, Δ are morphism variables in the categorical signature with types $0 \to 1$ and $1 \to 0$ respectively. This is just Thm. 3.12 from [18], which is restating the classic result from [14]. For frame-point graphs this is Thm. 5.5.10 in [15].

Note that this does not make the semantics categorical, since the *link* construction is not defined categorically but only concretely. However, any term (with our without link) corresponds to a diagram which can be described categorically. An immediate consequence of Prop. 2 is that all derivations of a well typed term produce the same uniflow diagram, possibly with a different anchoring relation.

Lemma 3. *The type system of uniflow diagrams is coherent up to the anchoring relation.*

Note that different derivations may require different anchoring relations, even though the underlying diagrams are equal as FPGs.

Lemma 4 (Definability)**.** *Any uniflow diagram is definable just in terms of composition, tensor, link.*

Proof. We introduce exactly two link nodes on each connector (a wire homeomorphism) and we label them with fresh variable names. Suppose there are j wires and k boxes, m inputs and n outputs. Let us denote by u_i where $i = 1,n$ the labels associated with input connectors and by v_i where $i = 1,m$ the labels associated with the output connectors. Let $y_{i,j}$ (respectively $z_{i,j}$) be the labels of the jth input (respectively output) for each constant occurrence k_i. Let $\vec{u} = \bigotimes_{i=1,n} u_i, \vec{v} = \bigotimes_{i=1,m} v_i$, $\vec{y_i} = \bigotimes_{j=1,\mathrm{dom}(k_i)} y_{i,j}, \vec{z_i} = \bigotimes_{j=1,\mathrm{codom}(k_i)} z_{i,j}$. Let $M_0 = \vec{u}; \left(\bigotimes_{i=1,k} \vec{y_i}; k_i; \vec{z_i}\right); \vec{v}$. The term is $M = \overline{x_1 x_1'} \cdots \overline{x_n x_n'}.M$. A link $\overline{x_i x_i'}$ is created if there is an edge between the nodes labelled by x_i and x_i' in M_0, with x_i, x_i' chosen from the variables introduced above. M should be a closed form. It is straightforward to show that this is indeed denotes the desired diagram. We only need to prove that this is well typed. The subterm $\bigotimes_{i=1,k} \vec{y_i}; f_i; \vec{z_i}$ is clearly well typed. Moreover, we can always extend the order with any $x_i \leq x_i'$ before linking without creating cycles because the starting graph is a DAG. □

The construction in the proof is just the flat nominal notation used by conventional graph languages.

Theorem 5. *For any term M there exist terms $\tilde{M}$ and $\hat{M}$ such that $\tilde{M}$ is link-free and $\hat{M}$ is free of structural combinators (identity, symmetry) and $[\![M]\!] = [\![\tilde{M}]\!] = [\![\hat{M}]\!]$.*

Example 1. *Symmetry at any type, e.g. $\gamma_{2,2} : 4 \to 4$ can be defined in combinatorial or nominal style:*

$$\gamma_{2,2} = \overline{aa'}.\overline{bb'}.\overline{cc'}.\overline{dd'}.a \otimes b \otimes c \otimes d \otimes c' \otimes d' \otimes a' \otimes b' = (i \otimes \gamma \otimes i); (\gamma \otimes \gamma); (i \otimes \gamma \otimes i).$$

Any $\gamma_{m,n}$ for $m,n \neq 0$ can be defined from γ, and identity id_m at any non-zero type can be defined from i. We sometimes write the identity id_m as just m.

Example 2. *We can now revisit our introductory example. The most succinct description mixes variables and structural connectors: $\overline{x'x}.\overline{yy'}.(f \otimes y); (i \otimes x' \otimes y' \otimes i); (g \otimes x)$.*

It can be easily checked that all equational properties of the underlying PROP are carried over to the syntax. We interpret $\Gamma \mid \Delta \vdash M \equiv N$ as the equality of the uniflow diagrams represented by $[\![\Gamma \mid \Delta \vdash M]\!]$ and $[\![\Gamma \mid \Delta \vdash N]\!]$ (ignoring the anchoring order). The equations are now parametrised by the set of free variables, as in Fig. 3, assuming integers m,n, etc. are chosen so that the terms are well-typed.

$$\Gamma \mid \Delta \vdash (M_1;M_2);M_3 \equiv M_1;(M_2;M_3) \qquad \text{associativity of composition} \qquad (2)$$
$$\Gamma \mid \Delta \vdash id_m;M \equiv M;id_n \equiv M \qquad \text{identity} \qquad (3)$$
$$\Gamma \mid \Delta \vdash (M;M') \otimes (N;N') \equiv (M \otimes N);(M' \otimes N') \qquad \text{tensor functoriality} \qquad (4)$$
$$\Gamma \mid \Delta \vdash (M_1 \otimes M_2) \otimes M_3 \equiv M_1 \otimes (M_2 \otimes M_3) \qquad \text{strictness} \qquad (5)$$
$$\Gamma \mid \Delta \vdash M \otimes 0 \equiv 0 \otimes M \equiv M \qquad \text{strictness} \qquad (6)$$
$$\Gamma \mid \Delta \vdash (M_1 \otimes M_2);\gamma_{n_1,n_2} \equiv \gamma_{m_2,m_1};(M_2 \otimes M_1) \qquad \text{symmetry} \qquad (7)$$
$$\vdash \gamma_{0,1} \equiv \gamma_{1,0} \equiv 1 \qquad \text{strict symmetry} \qquad (8)$$
$$\vdash \gamma_{m,n+p} \equiv (\gamma_{m,n} \otimes p);(n \otimes \gamma_{m,p}) \qquad \text{strict symmetry} \qquad (9)$$

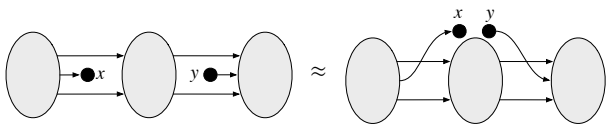
$$\Gamma \mid \Delta \vdash \overline{xy}.M \equiv \overline{x'y'}.M\{x'y'/xy\} \quad x',y' \text{ fresh} \qquad \text{alpha-equivalence} \qquad (10)$$
$$\Gamma \mid \Delta \vdash \overline{xy}.(M;N) \equiv (\overline{xy}.M);N \quad x,y \notin fv(N) \qquad \text{scope extrusion} \qquad (11)$$
$$\Gamma \mid \Delta \vdash \overline{xy}.(M \otimes N) \equiv (\overline{xy}.M) \otimes N \quad x,y \notin fv(N) \qquad \text{scope extrusion} \qquad (12)$$
$$\Gamma \mid \Delta \vdash \overline{xy}.(M;N) \equiv M;\overline{xy}.N, \quad x,y \notin fv(M) \qquad \text{scope extrusion} \qquad (13)$$
$$\Gamma \mid \Delta \vdash \overline{xy}.(M \otimes N) \equiv M \otimes \overline{xy}.N, \quad x,y \notin fv(M) \qquad \text{scope extrusion} \qquad (14)$$
$$\vdash \overline{xy}.x;y \equiv 1 \qquad \text{link identity} \qquad (15)$$

Figure 3: Equational theory of uniflow diagrams

We can now turn our attention to the equational theory of uniflow diagrams (Fig. 3). Since $\overline{xy}.M$ is a binding construct, we expect alpha-equivalence to hold, which is reflected in a new axiom, Eqn. 10. Variable renaming $M\{x/y\}$ and the set of free variables $fv(M)$ are defined in the obvious way. For name management we also have new equations for scope extrusion (Eqn. 11–14). Finally, the connection between variables and the structural connectors, namely identity, is given in Eq. 15.

Theorem 6. *The equational theory of uniflow diagrams is sound and complete for the given semantics.*

Proof. The soundness is immediate. For completeness, consider two terms such that $[\![\Gamma \mid \Delta \vdash M]\!] = [\![\Gamma \mid \Delta \vdash N]\!]$. We first rename all bound variables using alpha-equivalence, then, using Eqns. 11 and 13 we move all binders into global scope. In case the resulting terms are link-free the equation $\Gamma \mid \Delta \vdash M \equiv N$ is provable because the terms denote morphism in the free strict symmetric monoidal category over signature K extended with the uninterpreted variables Γ, Δ (Prop. 2). Suppose that one of the terms has an outermost link, so it has form $\Gamma \mid \Delta \vdash \overline{xy}.M$. We reason diagrammatically about term $\Gamma, x \mid \Delta, y \vdash M$, which is a morphism in the free SMC over K extended with $\Gamma \uplus \Delta \uplus \{x,y\}$. We can always redraw the diagram to an isomorphic diagram in which the nodes labelled by x and y are adjacent:



This diagrammatic equivalence can be proved in the equational theory of the uniflow diagrams, noting that $\gamma_{m,n}$ and the identities at m can be defined in terms of γ and i. Moreover, the diagram on the right will correspond to a term which has subterm $x;y$, with variables x,y not occurring anywhere else.

$$\frac{}{x \mid - \vdash x : 0 \to 1} \qquad \frac{}{- \mid x, \vdash x : 1 \to 0} \qquad \frac{}{- \mid - \vdash i : 1 \to 1} \qquad \frac{}{- \mid - \vdash \gamma : 2 \to 2}$$

$$\frac{\Gamma \mid \Delta \vdash M : m_1 \to m_2 \qquad \Gamma' \mid \Delta' \vdash N : n_1 \to n_2}{\Gamma \uplus \Gamma' \mid \Delta \uplus \Delta' \vdash M ; N : m_1 \to n_2}\, m_2 = n_1 \qquad \frac{\Gamma \mid \Delta \vdash M : m_1 \to m_2 \qquad \Gamma' \mid \Delta' \vdash N : n_1 \to n_2}{\Gamma \uplus \Gamma' \mid \Delta \uplus \Delta' \vdash M \otimes N : m_1 \to n_2}$$

$$\frac{\Gamma \mid \Delta \vdash M : m_1 \to m_2}{\Gamma \mid \Delta \vdash \mathrm{Tr}^i(M) : m_1 - i \to m_2 - i} \qquad \frac{\Gamma, x \mid \Delta, y \vdash M : m_1 \to m_2}{\Gamma \mid \Delta \vdash \overline{xy}.M : m_1 \to m_2}$$

Figure 4: Type system for biflow diagrams

So we can apply Eqns. 11 and 13 repeatedly until the link binder only binds this subterm, $\overline{xy}.x;y$. Then we use Eqn. 15 to replace this subterm with the identity. We repeat the process until all binders are eliminated. □

3 Biflow diagrams

Let us consider now strict symmetric *traced* monoidal categories and their associated diagrammatic language, *biflow diagrams* [7]. Biflow diagrams are labelled directed graphs with vertices V and edges E, equipped with a partial injection $f : V \rightharpoonup L$ from vertices to labels. Biflow diagrams are labelled directed graphs with interfaces I, O, lists of unlabelled ports, equivalent up to vertex renaming and wire homeomorphism. In other words, biflow diagrams are uniflow diagrams minus the anchoring relation, similar to Kissinger's *FPGs*. The increased expressiveness of the diagrammatic language is reflected into a relaxation of the type system. The type judgements are $\Gamma \mid \Delta \vdash M : m_1 \to m_2$, similar to the uniflow case but without an anchoring relation. The rules are also similar, with the addition of new rules for *trace* (Fig. 4). If the trace is over the unit we omit the superscript. Traces for $i > 1$ need not be included as primitive, as they can be constructed out of unit traces.

The diagrammatic interpretation is the same as in the previous section, omitting the anchoring order on vertices. The interpretation of the unit trace operator is the standard one, a *feedback* edge between the top input and output ports. Assuming that $[\![\Gamma \mid \Delta \vdash M : m+1 \to n+1]\!] = (V, i :: I, o :: O, E, f)$, $[\![\Gamma \mid \Delta \vdash \mathrm{Tr}(M) : m \to n]\!] = (V, I, O, E \cup \{(o,i)\})$. For equational completeness we also include the (trivial) trace over 0, $\mathrm{Tr}^0(M) = M$. Soundness holds, immediately.

Lemma 7 (Biflow definability)**.** *Any biflow diagram is definable in terms of composition, tensor, link.*

Theorem 8. *For any term M in the language of biflow diagrams, there are forms $\tilde{M}$ and $\hat{M}$ such that $\tilde{M}$ is link-free and $\hat{M}$ is free of structural combinators (identity, symmetry, trace) and $[\![M]\!] = [\![\tilde{M}]\!] = [\![\hat{M}]\!]$.*

In the equational theory we inherit all the equations from the previous section, plus the trace equations [12]. All the additional equations are given in Fig. 5. Surprisingly, we do not need new equations for link, except for scope extrusion in the presence of trace (Eqn. 23).

Other relations between trace and link can be derived in the existing equational theory. The proposition below is clearly sound in the model, corresponding to the two alternative descriptions of a back-link, using trace or *link*.

Proposition 9. $\Gamma \mid \Delta \vdash \mathrm{Tr}(M) \equiv \overline{yx}.(x \otimes m); M; (y \otimes n)$

Proof.

$$\begin{aligned} \overline{yx}.(x \otimes m); M; (y \otimes n) &= \overline{yx}.\mathrm{Tr}^0\big((x \otimes m); M; (y \otimes n)\big) && \text{(vanishing)} \\ &= \overline{yx}.\mathrm{Tr}\big(m; M; (y; x \otimes n)\big) && \text{(sliding)} \end{aligned}$$

$$\Gamma \mid \Delta \vdash \mathrm{Tr}^m((g \otimes n); f) \equiv g; \mathrm{Tr}^m(f) \qquad \text{left-tightening} \qquad (16)$$
$$\Gamma \mid \Delta \vdash \mathrm{Tr}^m(f; (g \otimes n)) \equiv \mathrm{Tr}^m(f); g \qquad \text{right-tightening} \qquad (17)$$
$$\Gamma \mid \Delta \vdash \mathrm{Tr}^m(f; (n \otimes g)) \equiv \mathrm{Tr}^p((q \otimes g); f) \qquad \text{sliding} \qquad (18)$$
$$\Gamma \mid \Delta \vdash \mathrm{Tr}^0(f) \equiv f \qquad \text{vanishing} \qquad (19)$$
$$\Gamma \mid \Delta \vdash \mathrm{Tr}^{m+n}(f) \equiv \mathrm{Tr}^m(\mathrm{Tr}^n(f)) \qquad \text{vanishing} \qquad (20)$$
$$\Gamma \mid \Delta \vdash \mathrm{Tr}^m(n \otimes s) \equiv n \otimes \mathrm{Tr}^m(f) \qquad \text{superposing} \qquad (21)$$
$$\vdash \mathrm{Tr}^1(\gamma) \equiv 1 \qquad \text{yanking} \qquad (22)$$
$$\Gamma \mid \Delta \vdash \overline{xy}.\mathrm{Tr}^i(M) \equiv \mathrm{Tr}^i(\overline{xy}.M), \quad i \in \{0,1\}. \qquad \text{scope extrusion} \qquad (23)$$

Figure 5: Additional equations for biflow diagrams

$$= \overline{yx}.\mathrm{Tr}\big(M; (y; x \otimes n)\big) \qquad \text{(identity)}$$
$$= \mathrm{Tr}\big(M; (\overline{yx}.(y; x) \otimes n)\big) \qquad \text{(link scope)}$$
$$= \mathrm{Tr}\big(M; (1 \otimes n)\big) \qquad \text{(link)}$$
$$= \mathrm{Tr}\big(M\big) \qquad \text{(identity)}$$

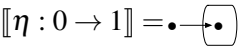
□

Theorem 10. *The equational theory of biflow diagram is sound and complete for the given semantics.*

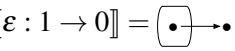
Note: Diagrams corresponding to compact-closed categories can be described analogously.

4 Monoid and comonoid structures

In the specification of diagrams corresponding to circuits and systems two kinds of structural components are used quite extensively: the splitting of two connectors and the joining of two connectors. A *monoid*, in this particular diagrammatic context is a pair of constant morphisms (ϕ, η) where $\phi : 2 \to 1$ is the *multiplication* and $\eta : 1 \to 0$ the *unit*, interpreted as the following biflow diagrams corresponding to "joining two connectors" and, respectively, to a "dangling output" connector:

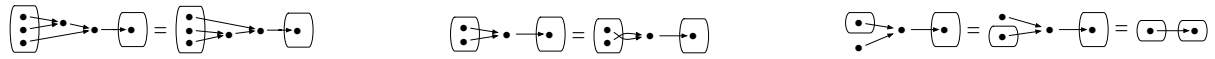
$[\![\phi : 2 \to 1]\!] =$  $[\![\eta : 0 \to 1]\!] =$  .

Conversely, a *co-monoid* consist of a *co-multiplication* $\psi : 1 \to 2$ corresponding to "splitting" two connectors and a *co-unit* $\varepsilon : 0 \to 1$, a "dangling input".

$[\![\psi : 1 \to 2]\!] =$  $[\![\varepsilon : 1 \to 0]\!] =$  .

The monoid should be associative, commutative and have a unit law. The corresponding diagrams are in Fig. 6. The co-monoid should be co-associative, co-commutative and have a co-unit, with the equations and diagrams the converse of the above.

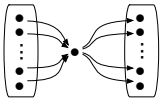
Other laws that apply in some contexts are *Frobenius* $((\phi \otimes 1); (1 \otimes \psi) = \phi; \psi)$ and the *special* law $(\psi; \phi = 1)$. Diagrammatically they are:



$(\phi \otimes 1);\phi = (1 \otimes \phi);\phi$ $\qquad$ $\phi = \gamma;\phi$ $\qquad$ $(1 \otimes \eta);\phi = (\eta \otimes 1);\phi = 1$

Figure 6: Commutative monoid laws

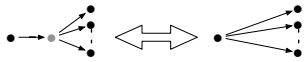
If all these equations are satisfied, any structure constructed out of the multiplication, co-multiplication, unit and co-unit can be reduced to a canonical form called informally a *spider diagram* [6]. Informally, all such constructions will denote a diagram of shape equivalent to



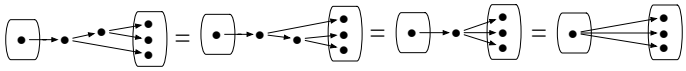
Precisely how these equations are chosen and how they are incorporated into the underlying graph model presents many choices that must be dealt with on a case-by-case basis, depending on the intended applications. In the next sequel we consider two interesting scenarios, but more variations are possible.

4.1 Comonoid structure

One natural way to add a comonoid structure to a biflow diagram is by adding constant morphisms $\psi : 1 \to 2$ and $\varepsilon : 1 \to 0$ with the semantics of the previous section. The wire-homeomorphism, which consists of the removal of all possible unlabelled nodes (or, conversely, the insertion of spurious unlabelled nodes) carry over in the obvious way to the new setting. Note that in the absence of the monoid structure all wire nodes, by construction, have exactly one incoming and one outgoing edge. With the monoid structure in place, every wire node has exactly one incoming edger and zero, one or more outgoing edges and wire homeomorphisms need to re-assign the target of the edge. The FPG $(V \uplus \{a\}, I, O, E, f)$ such that $f(a)$ is undefined (a is a wire node) is (still) homeomorphic to the FPG $(V, I, O, E \setminus \{a\} \cup \{(b,c) \mid (b,a),(a,c) \in E\})$. Diagrammatically, the homeomorphism is represented as:



The associativity, co-commutativity and co-unit laws now come then directly from the new wire homeomorphism. For example $\psi;(\psi \otimes 1) = \psi;(1 \otimes \psi)$ is:



To reflect the fact that wire nodes have now exactly one incoming and zero or more outgoing edges we update the type system to allow contraction and weakening for input variables while output variables preserve the linear discipline, as seen if Fig. 7. Even though the semantics of link remains the same, the graph invariants are different because of the presence of the co-multiplication and the co-unit. The diagram for $\overline{xy}.M$ is still the connection of (the single) occurrence of the node labelled with x to the (zero or more) occurrences of the node(s) labelled with y, followed by the removal of the labels. The mathematical formulation is the same as before (Eqn. 1), except for the anchoring relation.

$$\frac{}{\Gamma,x \mid - \vdash x : 0 \to 1} \qquad \frac{}{\Gamma \mid x \vdash x : 1 \to 0} \qquad \frac{}{\Gamma \mid - \vdash 1 : 1 \to 1}$$

$$\frac{}{\Gamma \mid - \vdash \gamma : 2 \to 2} \qquad \frac{}{\Gamma \mid - \vdash \psi : 1 \to 2} \qquad \frac{}{\Gamma \mid - \vdash \varepsilon : 1 \to 0}$$

$$\frac{\Gamma \mid \Delta \vdash M : m_1 \to m_2 \qquad \Gamma \mid \Delta' \vdash N : n_1 \to n_2}{\Gamma \mid \Delta \uplus \Delta' \vdash M;N : m_1 \to n_2}\ m_2 = n_1 \qquad \frac{\Gamma \mid \Delta \vdash M : m_1 \to m_2 \qquad \Gamma \mid \Delta' \vdash N : n_1 \to n_2}{\Gamma \mid \Delta \uplus \Delta' \vdash M \otimes N : m_1 \to n_2}$$

$$\frac{\Gamma \mid \Delta \vdash M : m_1 \to m_2 \qquad i = 0,1}{\Gamma \mid \Delta \vdash \mathrm{Tr}^i(M) : m_1 - i \to m_2 - i} \qquad \frac{\Gamma,x \mid \Delta,y \vdash M : m_1 \to m_2}{\Gamma \mid \Delta \vdash \overline{xy}.M : m_1 \to m_2}$$

Figure 7: Type system for diagrams with a co-monoid.

$$\frac{}{\Gamma,x \mid \Delta \vdash x : 0 \to 1} \qquad \frac{}{\Gamma \mid x,\Delta \vdash x : 1 \to 0} \qquad \frac{}{\Gamma \mid \Delta \vdash 1 : 1 \to 1} \qquad \frac{}{\Gamma \mid \Delta \vdash \phi : 2 \to 1}$$

$$\frac{}{\Gamma \mid \Delta \vdash \gamma : 2 \to 2} \qquad \frac{}{\Gamma \mid \Delta \vdash \psi : 1 \to 2} \qquad \frac{}{\Gamma \mid \Delta \vdash \varepsilon : 1 \to 0} \qquad \frac{}{\Gamma \mid \Delta \vdash \eta : 0 \to 1}$$

$$\frac{\Gamma \mid \Delta \vdash M : m_1 \to m_2 \qquad \Gamma \mid \Delta \vdash N : n_1 \to n_2}{\Gamma \mid \Delta \vdash M;N : m_1 \to n_2}\ m_2 = n_1 \qquad \frac{\Gamma,x \mid \Delta,y \vdash M : m_1 \to m_2}{\Gamma \mid \Delta \vdash \overline{xy}.M : m_1 \to m_2}$$

$$\frac{\Gamma \mid \Delta \vdash M : m_1 \to m_2 \qquad \Gamma \mid \Delta \vdash N : n_1 \to n_2}{\Gamma \mid \Delta \vdash M \otimes N : m_1 \to n_2} \qquad \frac{\Gamma \mid \Delta \vdash M : m_1 \to m_2 \qquad i = 0,1}{\Gamma \mid \Delta \vdash \mathrm{Tr}^i(M) : m_1 - i \to m_2 - i}$$

Figure 8: Type system for biflow Frobenius diagrams

Lemma 11 (Biflow comonoid definability)**.** *Any biflow diagram with a comonoid structure is definable in terms of composition, tensor and link.*

From this it follows immediately that the *link* construct is enough in terms of expressiveness, making the explicit use of the co-unit and the co-multiplication optional.

Theorem 12. *For any term M in the language of biflow diagrams with comonoids, there are forms $\tilde{M}$ and $\hat{M}$ such that $\tilde{M}$ is link-free and $\hat{M}$ is free of structural combinators (identity, symmetry, trace, comultiplication, counit) and $[\![M]\!] = [\![\tilde{M}]\!] = [\![\hat{M}]\!]$.*

The equational theory of the diagrammatic language is extended by the following two new equations:

$$\boxed{\varepsilon = \overline{xy}.x \qquad \psi = \overline{xy}.x \otimes y \otimes y.}$$

Theorem 13. *The equational theory of biflow diagrams with a comonoid is sound and complete for the given semantics.*

An analogous link syntax and equational theory for a monoid structure is obvious.

4.2 Frobenius structure

A particularly useful combination of the monoid and the comonoid structures uses the Frobenius and special axioms, leading to so-called *spider diagrams* [6]. Syntactically, besides the new constants ϕ and η, the monoid multiplication and unit, the type judgements now lose all linearity, with link nodes useable zero, one or more times both as input and output (Fig. 8).

Theorem 14. *For any term M in the language of biflow diagrams with a Frobenius structure, there are forms $\tilde{M}$ and $\hat{M}$ such that $\tilde{M}$ is link-free and $\hat{M}$ is free of structural combinators (identity, symmetry, trace, (co)multiplication, (co)unit) and $[\![M]\!] = [\![\tilde{M}]\!] = [\![\hat{M}]\!]$.*

The two equations for the monoid structure are: $\boxed{\eta = \overline{xy}.y, \quad \phi = \overline{xy}.x \otimes x \otimes y.}$

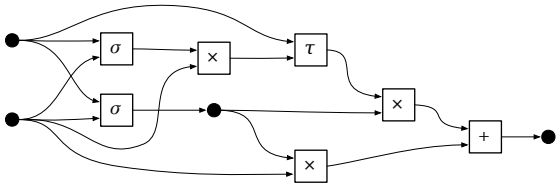
Theorem 15. *The equational theory of biflow diagrams with a Frobenius structure is sound and complete for the given semantics.*

Note that the interaction of the monoid and co-monoid is not necessarily subject to the Frobenius law in diagrammatic languages. This is not a structure that is well studied in its own right, but it turns out to occur naturally in the treatment of digital circuits [10]. The syntax in this case is less natural and the equational properties are more elusive, but the expressiveness and convenience of the notation remains an important attribute.

5 Example

In this section we are giving an example of circuit-like system which has both structural features and arbitrary connections, namely a neural network. We will employ a free use of all the methods of specification available (TMC with monoid and co-monoid and the link operation) to give a succinct and informative description which is, in our subjective opinion, preferable to both the purely structural or the purely relational descriptions. These will be left as reader exercises.

The net we use as an example is the *gated recurrent unit* [5], a simplification of the popular *long short-term memory* net [13] which is broadly used in wide range of applications, from machine translation to image classification. The basic cell of the net has the following architecture, where σ, τ are activation functions and $\times, +$ arithmetic operations.

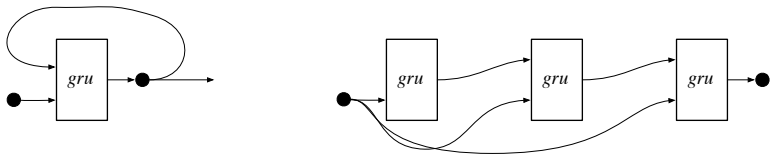


Let $\psi_2 : 2 \to 4$ be the (definable) co-monoid on a pair of inputs. The mixed-syntax and the purely structural notations for the GRU cell are:

$$\begin{aligned} gru &= \overline{x'x}.\overline{h'h}.(x' \otimes h'); (x \otimes h); \psi_2; (\sigma \otimes h \otimes \sigma); (x \otimes \times \otimes \psi); (\tau \otimes 1 \otimes \times); (\times \otimes 1); + \\ &= (\psi \otimes 1); (1 \otimes \psi \otimes \psi); (3 \otimes \psi \otimes \psi); (2 \otimes \gamma \otimes 3); (1 \otimes \sigma \otimes \sigma \otimes 2); (2 \otimes \gamma \otimes 1); \\ &\quad (1 \otimes \times \otimes \psi \otimes 1); (\tau \otimes 1 \otimes \times); (\times \otimes 1); + \end{aligned}$$

Note that in the above the x, h variables can have any width, not necessarily unit.

A recurrent neural net consists of GRU cell with a recurrent link. For practical application, instead of a recurrent link, a finite unfolding of the net is often used:



The two nets can be described as $\mathrm{Tr}(gru; \psi)$ and $\overline{x'x}.x'; (gru \otimes x); (gru \otimes x); gru$, respectively.

6 Related and further work

We have shown how variables can be used within a structural framework for diagram syntax. We showed how from the point of view of expressiveness the structural and the nominal syntax can be equivalent and complementary, if the use of variables is constrained by a suitable type system. We have highlighted a neat correspondence between non-linearity and the presence of a monoid or co-monoid structure in the diagram. By extending the structural notation with the new *link* construct we showed that the existing equational properties are preserved and, moreover, this new structure itself has good equational properties. We believe that the combination of structural and nominal syntax can sometimes improve the readability and usability of diagram languages. Although this paper is only concerned with syntax, it is part of a larger research effort aimed at creating hardware languages more similar to programming languages in terms of their categorical [9] and operational semantics [10]. As further work we intend to create *structural* conservative extensions of diagram languages such as VHDL, SIMULINK or DOT.

Several theoretical aspects we have not studied, but remain subject for further work. The most intriguing perhaps is the categorical semantics for the *link* structure itself. Similar diagrammatic structures have been conjectured in the study of higher-order π-calculus, with the input and output links modelled as adjoint profunctors and a link-like operation modelled as their composition [19]. A smaller issue, but practically important, is understanding the syntax and the axiomatisation of links for diagrams with monoid and comonoid structures in the absence of the Frobenius law. These structures occur in diagrams modelling systems with directed flow of information, such as digital circuits, which are well motivated by applications. Finally, the various definability results (e.g. Lem. 4) suggest the possibility of arriving at normal forms for various classes of diagrams.

In terms of the graph semantics, the frame point graphs is not the only possible starting point. Cospans of graphs [16] or hypergraphs [3] have also been used as semantics for diagrammatic monoidal categories. These alternative presentations may arguably offer, to the theorist, more conceptual clarity. What we find attractive about FPGs is the fact that they can be presented in a concrete way, eliding much of the categorical apparatus. We believe this could make our work accessible to a broader audience. Moreover, the *link* construct is particularly easy to interpret in FPGs. However, these alternative semantic frameworks along with categorical semantics are being investigated.

Acknowledgement: This work was supported in part by EPSRC grant EP/P004490/1.

References

[1] Samson Abramsky & Bob Coecke (2004): *A Categorical Semantics of Quantum Protocols*. In: *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*, pp. 415–425, doi:10.1109/LICS.2004.1319636.

[2] Per Bjesse, Koen Claessen, Mary Sheeran & Satnam Singh (1998): *Lava: Hardware Design in Haskell*. In: *Proceedings of the third ACM SIGPLAN International Conference on Functional Programming (ICFP '98), Baltimore, Maryland, USA, September 27-29, 1998.*, pp. 174–184, doi:10.1145/289423.289440.

[3] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Pawel Sobocinski & Fabio Zanasi (2016): *Rewriting modulo symmetric monoidal structure*. In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pp. 710–719, doi:10.1145/2933575.2935316.

[4] Filippo Bonchi, Pawel Sobocinski & Fabio Zanasi (2015): *Full Abstraction for Signal Flow Graphs*. In: *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pp. 515–526, doi:10.1145/2676726.2676993.

[5] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk & Yoshua Bengio (2014): *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1724–1734. Available at `http://aclweb.org/anthology/D/D14/D14-1179.pdf`.

[6] Bob Coecke & Ross Duncan (2008): *Interacting Quantum Observables*. In: *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, pp. 298–310, doi:10.1007/978-3-540-70583-3_25.

[7] V. E. Căzănescu & Gheorghe Stefănescu (1990): *Towards a New Algebraic Foundation of Flowchart Scheme Theory*. *Fundam. Inf.* 13(2), pp. 171–210. Available at `http://dl.acm.org/citation.cfm?id=97367.97373`.

[8] Dan R Ghica (2013): *Diagrammatic reasoning for delay-insensitive asynchronous circuits*. In: *Computation, Logic, Games, and Quantum Foundations*, Springer, pp. 52–68, doi:10.1007/978-3-642-38164-5_5.

[9] Dan R. Ghica & Achim Jung (2016): *Categorical semantics of digital circuits*. In Ruzica Piskac & Muralidhar Talupur, editors: *Formal Methods in Computer-Aided Design (FMCAD), 2016, Mountain View, California, USA*, pp. 41–49, doi:10.1109/FMCAD.2016.7886659.

[10] Dan R. Ghica, Achim Jung & Aliaume Lopez (2017): *Diagrammatic Semantics for Digital Circuits*. In Valentin Goranko & Mads Dam, editors: *26th EACSL Annual Conference on Computer Science Logic (CSL 2017), Leibniz International Proceedings in Informatics (LIPIcs)* 82, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 24:1–24:16, doi:10.4230/LIPIcs.CSL.2017.24.

[11] Philip Hackney & Marcy Robertson (2015): *On the Category of Props*. *Applied Categorical Structures* 23(4), pp. 543–573, doi:10.1007/s10485-014-9369-4.

[12] Masahito Hasegawa (1997): *Recursion from Cyclic Sharing: Traced Monoidal Categories and Models of Cyclic Lambda Calculi*. In: *Typed Lambda Calculi and Applications, Third International Conference on Typed Lambda Calculi and Applications, TLCA '97, Nancy, France, April 2-4, 1997, Proceedings*, pp. 196–213, doi:10.1007/3-540-62688-3_37.

[13] Sepp Hochreiter & Jürgen Schmidhuber (1997): *Long Short-Term Memory*. *Neural Computation* 9(8), pp. 1735–1780, doi:10.1162/neco.1997.9.8.1735.

[14] André Joyal & Ross Street (1991): *The geometry of tensor calculus, I*. *Advances in Mathematics* 88(1), pp. 55–112, doi:10.1016/0001-8708(91)90003-P.

[15] Aleks Kissinger (2012): *Pictures of processes: automated graph rewriting for monoidal categories and applications to quantum computing*. *arXiv preprint arXiv:1203.0202*. Available at `https://arxiv.org/abs/1203.0202`.

[16] Robert Rosebrugh, Nicoletta Sabadini & RFC Walters (2005): *Generic commutative separable algebras and cospans of graphs*. *Theory and applications of categories* 15(6), pp. 164–177. Available at `http://www.tac.mta.ca/tac/volumes/15/6/15-06abs.html`.

[17] Mehrnoosh Sadrzadeh, Stephen Clark & Bob Coecke (2013): *The Frobenius anatomy of word meanings I: subject and object relative pronouns*. *J. Log. Comput.* 23(6), pp. 1293–1317, doi:10.1093/logcom/ext044.

[18] Peter Selinger (2010): *A survey of graphical languages for monoidal categories*. In: *New structures for physics*, Springer, pp. 289–355, doi:10.1007/978-3-642-12821-9_4.

[19] J Vicary: Personal communication.