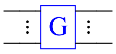
A Completeness proof: categories with finite products

The flow graphs we consider in this paper are finite directed edge-labelled, node-labelled graphs where the incoming and outgoing edges to each node are ordered, and each edge has exactly one start node, but multiple finish nodes. We impose a type discipline to ensure that the edge and node labels match. These graphs are the same as Hasegawa's *sharing graphs*.

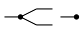
A *single-coloured flow graph* G over a signature Σ_V is:

- A finite set of edges (ranged over by E, F).
- A finite set of nodes (ranged over by N).
- For each edge, a label sort, written $E : X$.
- For each node, a label constructor, written $N : c$.
- A list of incoming edges and a list of outgoing edges, written $G : \mathbf{E} \to \mathbf{F}$.
- For each node, a list of incoming edges and a list of outgoing edges, written $N : \mathbf{E} \to \mathbf{F}$.
- If $N : \mathbf{E} \to \mathbf{F}$, $N : c$ and $c : \mathbf{X} \to \mathbf{Y}$, then $\mathbf{E} : \mathbf{X}$ and $\mathbf{F} : \mathbf{Y}$.
- Each edge occurs exactly once either as an incoming edge of the graph, or an outgoing edge of a node.

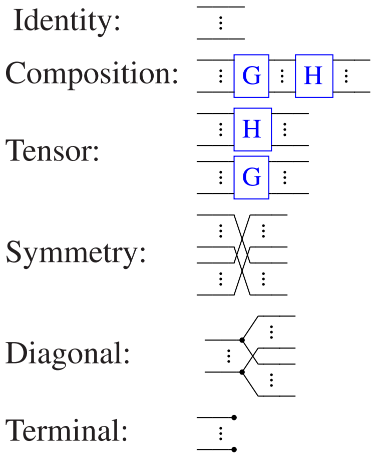
Such graphs can be drawn with incoming edges on the left and outgoing edges on the right:



Note that we insist that each edge has a unique source, but not a unique target. This allows edges to fork or terminate (but not the mirror-image graphs):



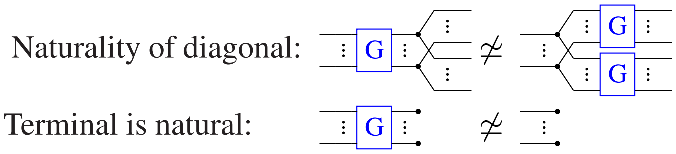
We can define the following operations on graphs (and below we shall show that they satisfy the equations necessary to be a category with finite products):



A morphism F between flow graphs G and G' is:

- A map from the edges of G to the edges of G'.
- A map from the nodes of G to the nodes of G'.
- The maps respect labels, incoming edges and outgoing edges.

Most of the axioms for a category with finite products are sound for flow graphs up to graph isomorphism, but the naturality conditions are *not* graph isomorphisms:



To find a graph model of finite products, we cannot consider graphs up to graph isomorphism. This is a familiar problem from concurrency theory, where graph isomorphism of labelled transition systems is too fine an equivalence, and so *bisimulation*, developed by Milner is used. We can adapt bisimulation to flow graphs, and below we show that the finite product axiomatization is sound and complete for flow graphs up to bisimulation.

A relation R between flow graphs G and G' is:

- A relation between the edges of G to the edges of G'.
- A relation between the nodes of G to the nodes of G'.
- The relations respect labels, incoming edges and outgoing edges:
  - if $E\,R\,E'$ and $E : X$ then $E' : X$,
  - if $N\,R\,N'$ and $N : \mathsf{c}$ then $N' : \mathsf{c}$,

– if $G : \mathbf{E} \to \mathbf{F}$ and $G' : \mathbf{E}' \to \mathbf{F}'$ then $\mathbf{E} \mathrel{R} \mathbf{E}'$ and $\mathbf{F} \mathrel{R} \mathbf{F}'$, and
– if $N : \mathbf{E} \to \mathbf{F}$ in G, $N' : \mathbf{E}' \to \mathbf{F}'$ in G' and $N \mathrel{R} N'$ then $\mathbf{E} \mathrel{R} \mathbf{E}'$ and $\mathbf{F} \mathrel{R} \mathbf{F}'$.

together with the symmetric conditions.

Note that (up to isomorphism) this definition is the same as requiring R to form a jointly monic span $G \leftarrow R \rightarrow G'$ in the category of flow graphs.

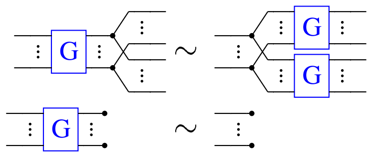
A relation R between flow graphs G and G' is a *simulation* iff:

- Whenever E is the nth outgoing edge of N in G and $E \mathrel{R} E'$ then E' is the nth outgoing edge of N' in G' and $N \mathrel{R} N'$.
- R is a function on incoming edges of G (that is for any incoming edge E of G there is precisely one edge E' of G' such that $E \mathrel{R} E'$).

A relation R between flow graphs G and G' is a *bisimulation* iff R and R^{-1} are simulations.

G and G' are *bisimilar* (written $G \sim G'$) iff there is a bisimulation between them.

Note that any graph isomorphism is a simulation, and so isomorphic graphs are bisimilar. The converse is not true, since:

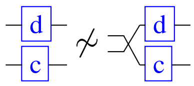


We are going to construct a category of graphs viewed up to bisimulation: in order for this definition to be valid, we require all of the categorical operations on graphs to respect bisimulation. Note that this is the reason for the (otherwise somewhat unnatural) second clause in the definition of simulation:

- R is a function on input edges (that is for any input edge E of G there is precisely one edge E' of G' such that $E \mathrel{R} E'$).

Without this clause, any relation between nodeless graphs would be a bisimulation, and in particular we could construct a bisimulation:

but:



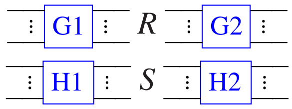
Since we have included the second clause in the definition of simulation, we have that nodeless graphs are bisimilar only when they are isomorphic, and so this counterexample fails.

Proposition (Bisimulation is a congruence). *Bisimulation is a congruence wrt the graph operations.*

Proof. It is routine to show that bisimulation is an equivalence.

To show that it is a congruence, we have to show it is respected by composition and tensor. Of these, tensor is simpler, so we shall show the case for composition.

If we have bisimulations R and S with:



we would like to find a bisimulation $R;S$ with:

Wlog, we shall assume the graphs have disjoint node and edge sets. Define union on graphs with overlapping edge sets as $G \cup G'$ with:

- Nodes and edges taken as the union of G and G's.
- Incoming edges from G.
- Outgoing edges from G'.

Given graph G including vector of edges $\mathbf{E}$, let $G[\mathbf{F}/\mathbf{E}]$ be the graph given by renaming edges $\mathbf{E}$ to $\mathbf{F}$.

We have graphs:

$$
\begin{aligned}
\mathrm{G1} &: \mathbf{E1} \to \mathbf{E1}' \\
\mathrm{G2} &: \mathbf{E2} \to \mathbf{E2}' \\
\mathrm{H1} &: \mathbf{F1} \to \mathbf{F1}' \\
\mathrm{H2} &: \mathbf{F2} \to \mathbf{F2}'
\end{aligned}
$$

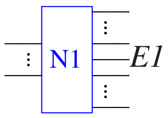
and:

$$\mathsf{G1;H1} = \mathsf{G1} \cup \mathsf{H1}[\mathbf{E1}'/\mathbf{F1}]$$
$$\mathsf{G2;H2} = \mathsf{G2} \cup \mathsf{H2}[\mathbf{E2}'/\mathbf{F2}]$$

Then define relation:

$$R;S\ R \cup S[\mathbf{E1}'/\mathbf{F1}]$$

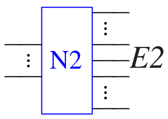
It is routine to show that this relation respects incoming and outgoing edges, labelling, and is an isomorphism between incoming edges. To show that $R;S$ is a bisimulation, consider any node in G1;H1:



where $E1\ R;S\ E2$. Then either:

- $E1 \in$ G1, so N1 $\in$ G1 and $E1\ R\ E2$. Since R is a bisimulation, we can find a N2 $\in$ G2 with N1 R N2.
- $E1 \in$ H1, so $E1$ is not an incoming edge of H1, so $E2$ is not an incoming edge of H2, so N1 $\in$ H1 and $E1\ S\ E2$. Since S is a bisimulation, we can find a N2 $\in$ H2 with N1 S N2.

In either case, we have found a node in G2;H2:



where $N1\ R;S\ N2$. Thus composition respects bisimulation. □

A category with finite products is *over* a signature Σ iff:

- For each sort X there is an object $[[X]]$.
- For each constructor $c : X_1 ... X_m \to Y_1 ... Y_n$ there is a morphism $[[c]] : [[X_1]] \times \cdots \times [[X_m]] \to [[Y_1]] \times \cdots \times [[Y_n]]$.

Acyclic flow graphs over a given signature Σ_V form a category Graph(Σ_V) over Σ_V where:

- Objects are vectors of sorts $\mathbf{X}$.

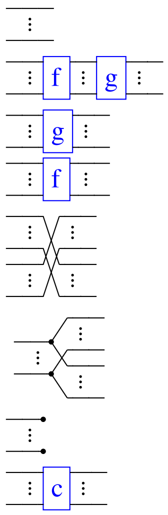
- Morphisms from $\mathbf{X}$ to $\mathbf{Y}$ are acyclic graphs $G : \mathbf{E} \to \mathbf{F}$ such that $\mathbf{E} : \mathbf{X}$ and $\mathbf{F} : \mathbf{Y}$, viewed up to bisimulation.

Proposition (Flow graphs form a category with finite products). *$\mathsf{Graph}(\Sigma_V)$ is a strict cartesian category over Σ_V.*

Proof. We have already defined the required operations, but we need to show they satisfy the axioms for a cartesian category. For this, we construct a bisimulation for each of the axioms. □

Since flow graphs over Σ_V form a cartesian category over Σ_V, there is a unique morphism [[_]] from the free cartesian category over Σ_V into $\mathsf{Graph}(\Sigma_V)$. We can define this syntactically as the term algebra (with the type system and axiomatization given for cartesian categories) $\mathsf{Term}(\Sigma_V)$:

$$
\begin{array}{rl}
\mathsf{f},\mathsf{g} ::= & \mathsf{id}_{\mathbf{X}} \\
\mid & \mathsf{f} \,;\, \mathsf{g} \\
\mid & \mathsf{f} \times \mathsf{g} \\
\mid & \mathsf{symm}_{\mathbf{X}\,\mathbf{Y}} \\
\mid & \mathsf{copy}_{\mathbf{X}} \\
\mid & \mathsf{discard}_{\mathbf{X}} \\
\mid & \mathsf{c}
\end{array}
$$



If terms can be proved equal using the axioms for a cartesian category we shall write $\vdash \mathsf{f} = \mathsf{g}$.

We would like to show soundness and completeness for this axiomatization, that is $\vdash \mathsf{f} = \mathsf{g}$ iff $[[\mathsf{f}]] \sim [[\mathsf{g}]]$. As is usual for such results, soundness is immediate, but completeness requires a normal form result. Most of the rest of this section is taken up by showing the required normalization results.

Proposition (Soundness). *If $\vdash f = g$ then $[[f]] \sim [[g]]$.*

Proof. Follows immediately from the fact that $\mathsf{Graph}(\Sigma_V)$ is a cartesian category. □

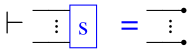
Proposition (Expressivity). *For any graph G, there is a term f such that $[[f]] \sim G$.*

Proof. Proved by induction on the number of outgoing edges of G. □

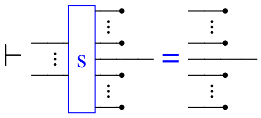
A *shuffle* (ranged over by s) is any term not including constructors c. A *permutation* (ranged over by p) is any shuffle not including copy or $\mathsf{discard}$.

Proposition (Completeness for shuffles). *If [[s1]] ∼ [[s2]] then* ⊢*s1 = s2*.

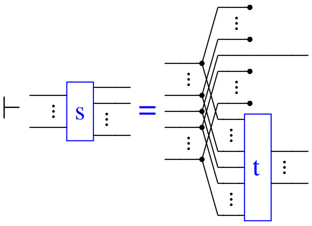
Proof. First show by induction on s that:



Then show by induction on s that:



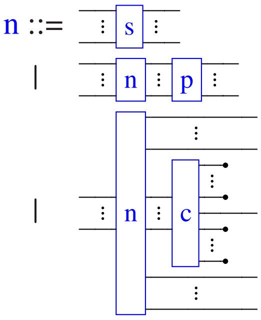
Then show by induction on s that we can find a shuffle t such that:



(The previous result is needed in the case of composition).

Finally, iteratively use the first and last properties to show that if [[s1]] ∼ [[s2]] then ⊢s1 = s2. □

A term is in *normal form* (ranged over by n) iff it is of the form:



Proposition (Normalization). *For any f we can find normal g such that* ⊢ *f = g*.

Proof. First show by induction on syntax that for any normal f1 and f2, we can find normal g such that ⊢f1 × f2 = g.

Then show by induction on syntax that for any normal f and shuffle s, we can find normal g such that ⊢f;s = g.

Then show by induction on syntax that for any normal f1 and f2, we can find normal g such that ⊢f1;f2 = g.

Finally show by induction on syntax that for any f, we can find normal g such that ⊢f = g. □

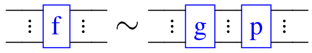
Define p_inv to be the permutation:

$$
\begin{aligned}
\mathrm{id\_inv} &= \mathrm{id} \\
(\mathrm{p}\ ;\ \mathrm{q})\mathrm{\_inv} &= \mathrm{q\_inv}\ ;\ \mathrm{p\_inv} \\
(\mathrm{p} \times \mathrm{q})\mathrm{\_inv} &= \mathrm{p\_inv} \times \mathrm{q\_inv} \\
\mathrm{symm\_inv} &= \mathrm{symm}
\end{aligned}
$$

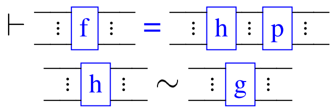
Proposition (Permutations are isos). *For any p, ⊢ p;p_inv = id and ⊢ p_inv;p = id.*

Proof. An induction on p. □

Proposition (Cancellation of permutations). *For any:*

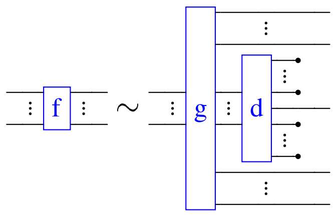


we can find h such that:

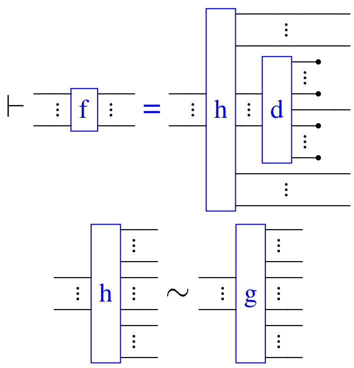


Proof. Let h be: f;p_inv. Then use soundness and p being an iso to show the required properties. □

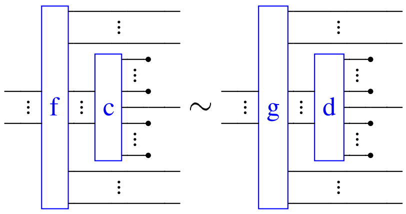
Proposition (Cancellation of nodes). *For any:*



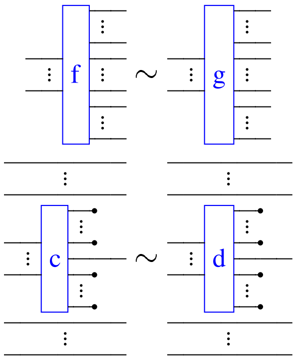
we can find h such that:



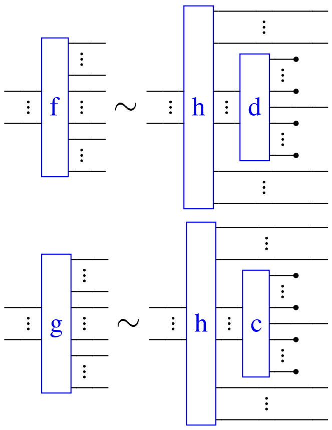
Proof. First, show a result about bisimulation on graphs; if:



then either:

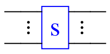


or we can find h such that:



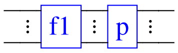
We now proceed by induction on f, which (by normalization) we can assume is in normal form. We have three cases:

- If f is:

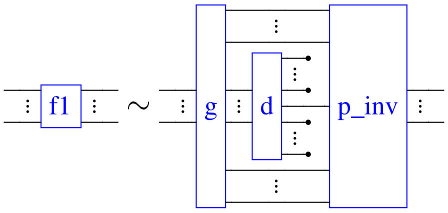


then we have a contradiction, since f is bisimilar to a graph with a node connected to an outgoing edge.

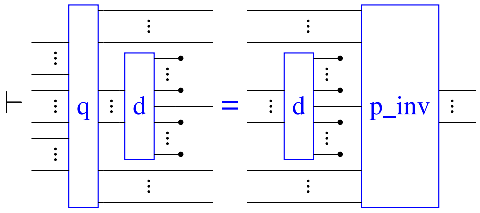
- If f is:



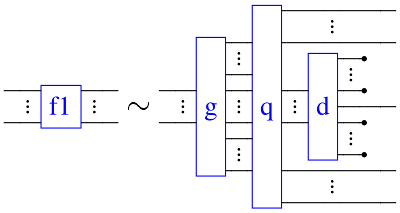
then:



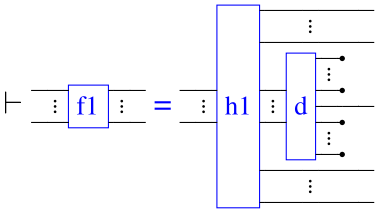
then find permutation q such that:

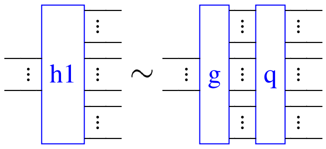


so we have:

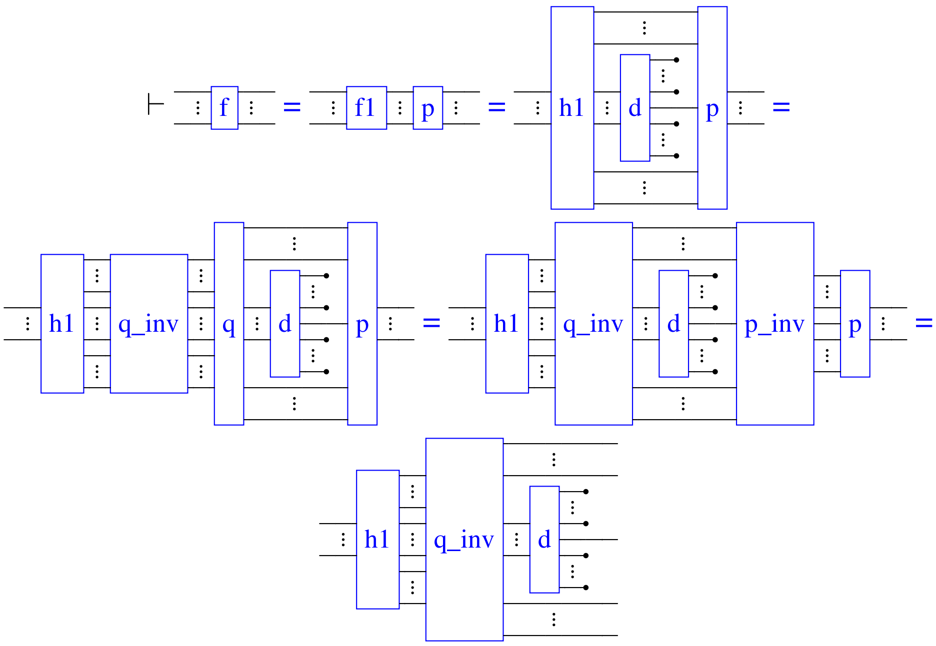


By induction, we can find h1 such that:

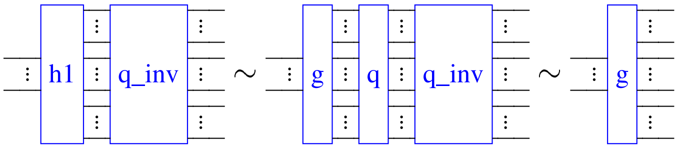




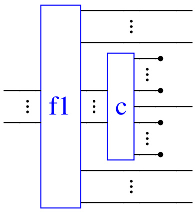
so:



and:

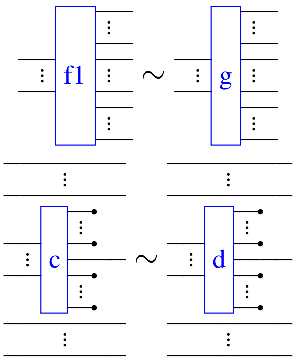


- If f is:



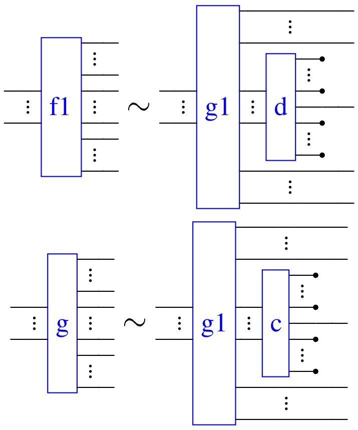
then by the above result about bisimulation either:

  – we have:

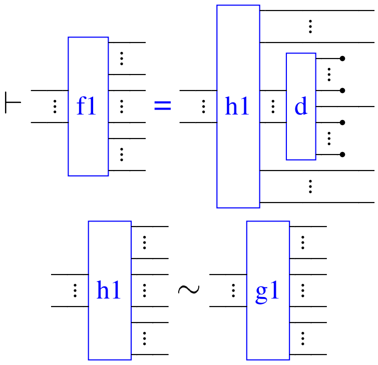


and we are finished,

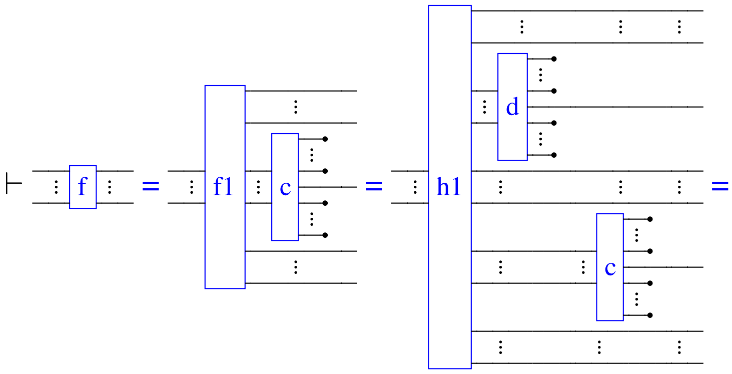
– or we can find g1 such that:

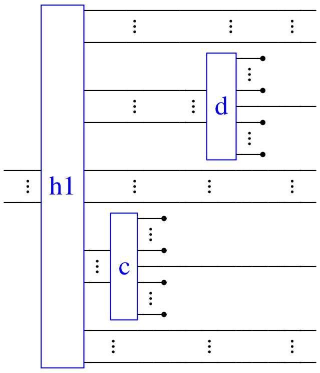


so by induction we can find h1 such that:

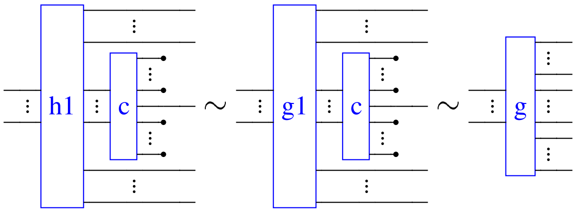


Then we have two cases depending on the relative positions of c and d, but the proofs are identical:





and:



□

Proposition (Completeness). *If* $[[f]] \sim [[g]]$ *then* $\vdash f = g$*.*

Proof. Find the normal form equal to g, and then cancel permutations and nodes iteratively, finishing with a use of completeness for shuffles, to prove $\vdash$f = g. □

Proposition (Initiality). *Graph(Σ_V) is the initial cartesian category over Σ_V.*

Proof. By construction, Term(Σ_V) is the initial cartesian category over Σ_V. Since Term(Σ_V) is expressive, we have a map term : Graph(Σ_V) $\rightarrow$ Term(Σ_V) with the property that [[term(G)]] $\sim G$. By soundness and completeness, this map is an isomorphism in the category of categories with finite products over Σ_V, and so Graph(Σ_V) is the initial cartesian category over Σ_V. □

B Completeness proof: symmetric monoidal categories

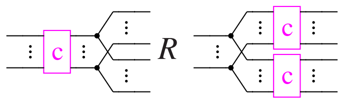
A *two-coloured flow graph* over two signatures Σ_V and Σ_C with the same sorts is a single-coloured flow graph over Σ_V except that:

- Each node is either labelled N : c or N : c.
- If $N : \mathbf{E} \rightarrow \mathbf{F}$, N : c and c : $\mathbf{X} \rightarrow \mathbf{Y}$, then $\mathbf{E} : \mathbf{X}$ and $\mathbf{F} : \mathbf{Y}$.

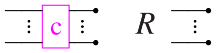
A simulation between two-coloured flow graphs is the same as for single-coloured flow graphs except that:

- R is a function on nodes labelled by Σ_C (that is for any node $N : \mathsf{c}$ of G there is precisely one node N' of G' such that $N\,R\,N'$).

For example, there is no bisimulation:



since if R is a function on central nodes, it must map the node labelled c to only one of the nodes on the right, and so R^{-1} is not a total function on central nodes. Similarly, there is no bisimulation:



since R does not map c to any node.

We can now replay the same proof of soundness and completeness as in the finite products case. In each case, the proof of the theorems is so similar to that in the finite products case that we elide it.

Let Graph(Σ_V,Σ_C) be the category of acyclic flow graphs over (Σ_V,Σ_C), viewed up to bisimulation.

- For graphs only containing val nodes, bisimulation in Graph(Σ_V,Σ_C) is the same as bisimulation in Graph(Σ_V).
- For graphs only containing central nodes, bisimulation in Graph(Σ_V,Σ_C) is the same as graph isomorphism.

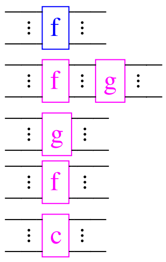
Proposition (Bisimulation is a congruence). *Bisimulation is a congruence wrt the graph operations.*

Proposition (Flow graphs form a strict symmetric monoidal category).

- *Graph(Σ_V,Σ_C) is a strict symmetric monoidal category over Σ_C.*
- *The inclusion Graph(Σ_V) $\hookrightarrow$ Graph(Σ_V,Σ_C) is an identity on objects symmetric monoidal functor.*

Define Term(Σ_V,Σ_C) as the free strict symmetric monoidal category over Σ_C which has Term(Σ_V) as a sub smc with the same objects.

$$
\begin{aligned}
f,g ::=\ & f \\
|\ & f\,;\,g \\
|\ & f \otimes g \\
|\ & c
\end{aligned}
$$



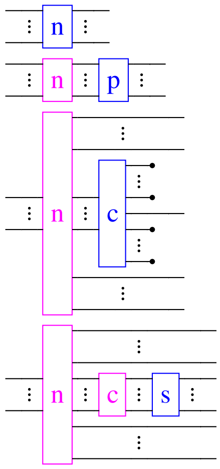
If terms can be proved equal using the axioms for an smc with a sub smc $\mathrm{Term}(\Sigma_V)$, we shall write $\vdash f = g$.

Proposition (Soundness). *If $\vdash f = g$ then $[[f]] \sim [[g]]$.*

Proposition (Expressivity). *For any graph G, there is a term f such that $[[f]] \sim G$.*

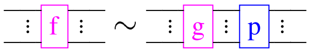
A term is in *normal form* (ranged over by n) iff it is of the form:

$$
\begin{aligned}
n ::=\ & \\
|\ & \\
|\ & \\
|\ &
\end{aligned}
$$

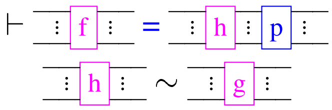


Proposition (Normalization). *For any f we can find normal g such that $\vdash f = g$.*

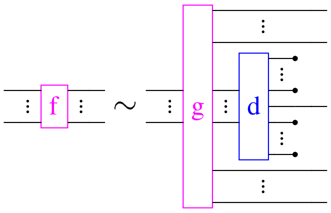
Proposition (Cancellation of permutations). *For any:*



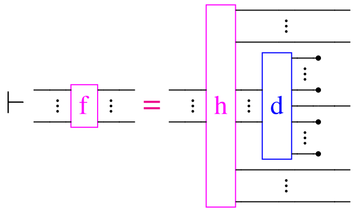
we can find h such that:

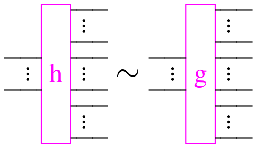


Proposition (Cancellation of val nodes). *For any:*

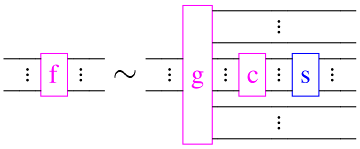


we can find h *such that:*

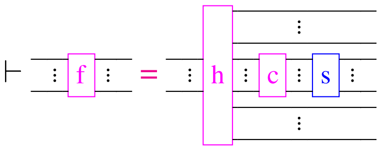


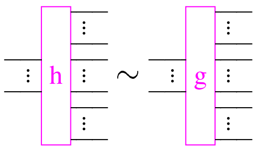


Proposition (Cancellation of central nodes). *For any:*



we can find h *such that:*





Proposition (Completeness). *If* $[[f]] \sim [[g]]$ *then* $\vdash f = g$*.*

Proposition (Initiality). *Graph*(Σ_V,Σ_C) *is the initial strict symmetric monoidal category over* Σ_C *with Graph*(Σ_V) *as a sub smc with the same objects.*

C Completeness proof: premonoidal categories

A *strict symmetric premonoidal category* is a category P together with:

- A strict symmetric monoidal category C with the same objects as P (called the *centre* of P).
- An identity on objects inclusion $\mathsf{C} \hookrightarrow \mathsf{P}$.
- Two functors:

$$\oslash : \mathsf{C} \times \mathsf{P} \to \mathsf{P}$$
$$\obslash : \mathsf{P} \times \mathsf{C} \to \mathsf{P}$$

  such that:

  – the three functors $\otimes$, $\oslash$ and $\obslash$ coincide on objects,
  – the three 'obvious' functors from $\mathsf{C} \times \mathsf{C}$ to P coincide, and
  – the symmetry in C is a natural isomorphism $X \oslash Y \simeq Y \obslash X$ in P.

This definition is based on Power's presentation of Power and Robinson's definition.

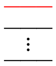
A *three-coloured flow graph* over three signatures Σ_V, Σ_C and Σ_P with the same sorts is a two-coloured flow graph over Σ_V and Σ_C except that:

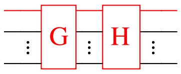
- Each edge is either labelled $E : X$ or $E : \mathsf{S}$.
- Each node is either labelled $N : \mathsf{c}$, $N : \mathsf{c}$ or $N : \mathsf{c}$.
- If $N : \mathbf{E} \to \mathbf{F}$, $N : \mathsf{c}$ and $\mathsf{c} : \mathbf{X} \to \mathbf{Y}$, then $\mathbf{E} : \mathbf{X}\mathsf{S}$ and $\mathbf{F} : \mathbf{Y}\mathsf{S}$.
- Each S-labelled edge occurs exactly once either as an outgoing edge of the graph, or an incoming edge of a node.

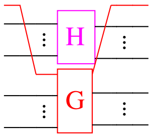
A simulation between three-coloured flow graphs is the same as for two-coloured flow graphs except that:

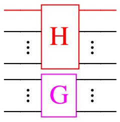
- R is a function on nodes labelled by Σ_P (that is for any node $N : \mathsf{c}$ of G there is precisely one node N' of G' such that $N \, R \, N'$).

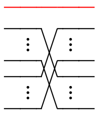
We can define the following operations on graphs (and below we shall show that they satisfy the equations necessary to be a premonoidal category):

Identity: 

Composition: 

Tensor left: 

Tensor right: 

Symmetry: 

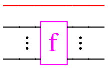
We can now replay the same proof of soundness and completeness as in the finite products case. In each case, the proof of the theorems is so similar to that in the finite products case that we elide it.

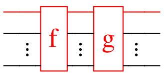
Let $\mathrm{Graph}(\Sigma_V,\Sigma_C,\Sigma_P)$ be the category of acyclic flow graphs over $(\Sigma_V,\Sigma_C,\Sigma_P)$ where the morphisms from $\mathbf{X}$ to $\mathbf{Y}$ are graphs from $\mathbf{X}S$ to $\mathbf{Y}S$, viewed up to bisimulation.

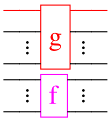
Proposition (Bisimulation is a congruence). *Bisimulation is a congruence wrt the graph operations.*

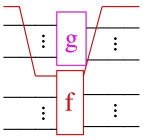
Proposition (Flow graphs form a premonoidal category). *$\mathit{Graph}(\Sigma_V,\Sigma_C,\Sigma_P)$ is a premonoidal category over Σ_P with centre $\mathit{Graph}(\Sigma_V,\Sigma_C)$.*

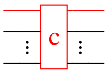
Define $\mathrm{Term}(\Sigma_V,\Sigma_C,\Sigma_P)$ as the free premonoidal category over Σ_P with central category $\mathrm{Term}(\Sigma_V,\Sigma_C)$.

$\mathrm{f},\mathrm{g} ::= \mathrm{f}$ 

$\mid\ \mathrm{f}\,;\mathrm{g}$ 

$\mid\ \mathrm{f} \oslash \mathrm{g}$ 

$\mid\ \mathrm{f} \obslash \mathrm{g}$ 

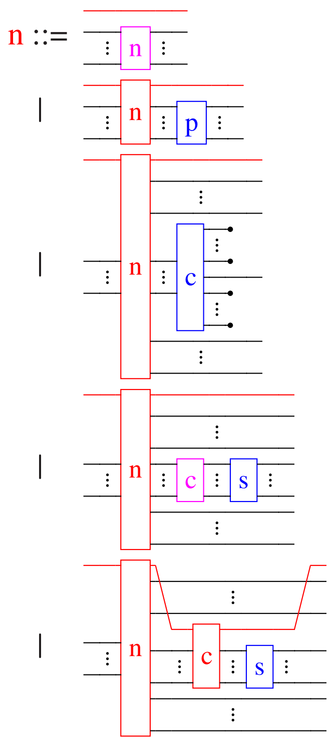
$\mid\ \mathrm{c}$ 

If terms can be proved equal using the axioms for a premonoidal category we shall write $\vdash \mathrm{f} = \mathrm{g}$.

Proposition (Soundness). *If* $\vdash f = g$ *then* $[[f]] \sim [[g]]$.

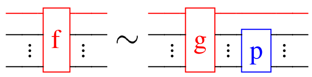
Proposition (Expressivity). *For any graph* G*, there is a term* f *such that* $[[f]] \sim G$.

A term is in *normal form* (ranged over by n) iff it is of the form:

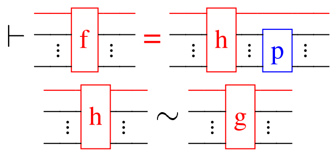


Proposition (Normalization). *For any* f *we can find normal* g *such that* $\vdash f = g$.

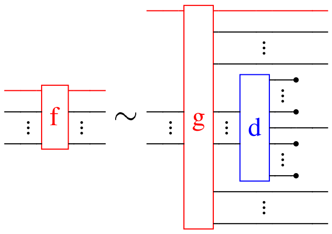
Proposition (Cancellation of permutations). *For any:*



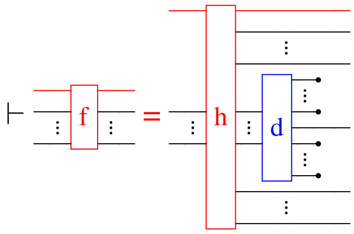
we can find h *such that:*

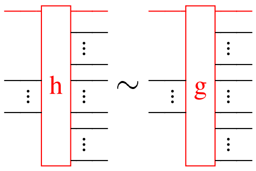


Proposition (Cancellation of val nodes). *For any:*

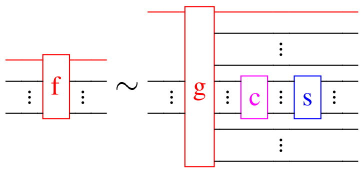


we can find h *such that:*

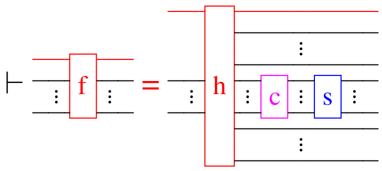


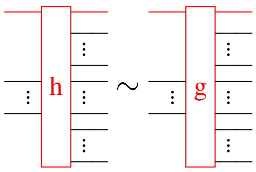


Proposition (Cancellation of central nodes). *For any:*

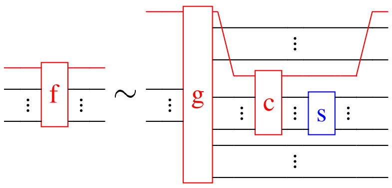


we can find h *such that:*

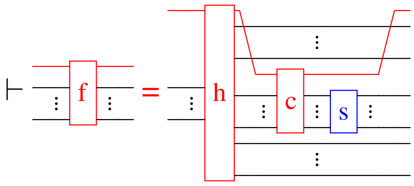


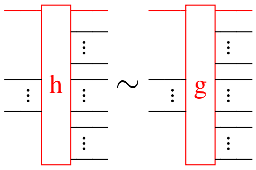


Proposition (Cancellation of proc nodes). *For any:*



we can find h such that:





Proposition (Completeness). *If* $[[f]] \sim [[g]]$ *then* $\vdash f = g$.

Proposition (Initiality). *Graph*$(\Sigma_V, \Sigma_C, \Sigma_P)$ *is the initial strict symmetric premonoidal category over* Σ_P *with Graph*(Σ_V, Σ_C) *as centre.*

D Completeness proof: partial traced cartesian categories

Given a strict symmetric monoidal category **C**: with a full subcategory (not necessarily symmetric monoidal) **TC**:

$$U : \mathbf{TC} \hookrightarrow \mathbf{C}$$

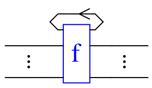
a *partial trace* is a natural transformation:

$$\mathrm{Tr}_{X\,Y\,A} : \mathbf{C}[X \otimes U(A), Y \otimes U(A)] \to \mathbf{C}[X, Y]$$

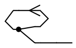
satisfying certain axioms (given below, since they are much simpler to present graphically than equationally).

As an example, let **TCpo** be the full subcategory of **Cpo** where all of the objects have a least element. Then **TCpo** $\hookrightarrow$ **Cpo** has a partial trace given by fixed points.

As another example, given a signature Σ_V where a subset of the sorts are define a cyclic single-coloured flow graph to be *traceable* iff any cyclic path goes through at least one edge labelled with a **traceable** sort. Taking **C** to be the category of traceable graphs up to bisimulation, and **TC** to be the full subcategory where the objects are traceable sorts, we have a partial trace given by forming a feedback loop:



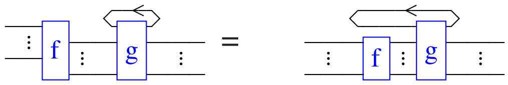
Note that in order to allow 'black holes':



we have to relax the condition that all edges have a unique source, and allow traceable edges to have at most one source. Then the above black hole is a graph with one outgoing edge with no source.

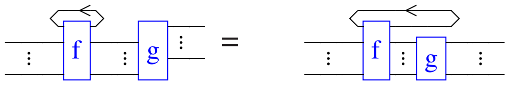
We shall provide a different axiomatization of traces than that provided by Joyal, Street and Verity, but we shall show below that in the case when **C** and **TC** are the same category, we recover their definition. Note that these axioms are all graph isomorphisms, and indeed are a complete axiomatization for graph isomorphism.

Naturality in X (for $\mathsf{f} : X' \to X$ and $\mathsf{g} : X \otimes U(A) \to Y \otimes U(A)$ in $\mathbf{C}$):



$$\mathsf{f};\, \mathrm{Tr}_{X\,Y\,A}\,(\mathsf{g}) \;=\; \mathrm{Tr}_{X'\,Y\,A}\,(\mathsf{f} \otimes \mathsf{id}_{U(A)};\, \mathsf{g})$$

Naturality in Y (for $\mathsf{f} : Y \to Y'$ and $\mathsf{g} : X \otimes U(A) \to Y \otimes U(A)$ in $\mathbf{C}$):

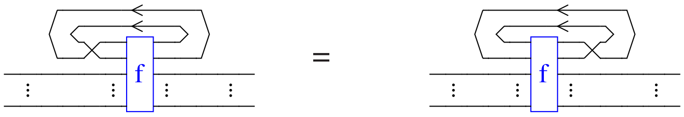


$$\mathrm{Tr}_{X\,Y\,A}\,(\mathsf{g});\, \mathsf{f} \;=\; \mathrm{Tr}_{X\,Y'\,A}\,(\mathsf{g};\, \mathsf{f} \otimes \mathsf{id}_{U(A)})$$

Yanking:

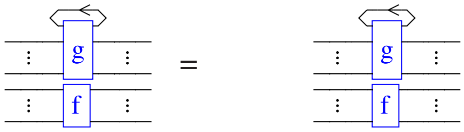
$$\mathrm{Tr}_{U(A)\,U(A)\,A}\,(\mathsf{symm}_{U(A)\,U(A)}) \;=\; \mathsf{id}_{U(A)}$$

Symmetry sliding (for $\mathsf{f} : X \otimes U(A) \otimes U(B) \to Y \otimes U(B) \otimes U(A)$):



$$\mathrm{Tr}_{X\,Y\,A}\,(\mathrm{Tr}_{(X \otimes U(A))\,(Y \otimes U(A))\,B}\,(\mathsf{f};\, (\mathsf{id}_Y \otimes \mathsf{symm}_{U(B)\,U(A)}))) \;=\; \mathrm{Tr}_{X\,Y\,B}\,(\mathrm{Tr}_{(X \otimes U(B))\,(Y \otimes U(B))\,A}\,((\mathsf{id}_X \otimes \mathsf{symm}_{U(B)\,U(A)});\, \mathsf{f}))$$

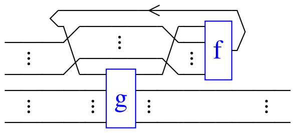
Strength (for $\mathsf{f} : V \to W$ and $\mathsf{g} : X \otimes U(A) \to Y \otimes U(A)$ in $\mathbf{C}$):



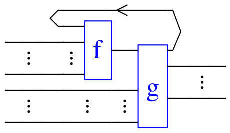
$$\mathsf{f} \otimes \mathrm{Tr}_{X\,Y\,A}\,(\mathsf{g}) \;=\; \mathrm{Tr}_{(V \otimes X)\,(W \otimes Y)\,A}\,(\mathsf{f} \otimes \mathsf{g})$$

For those readers familiar with Joyal, Street and Verity's axiomatization of a trace, the above axioms are obviously derivable from theirs. Moreover, the following lemmas show that we can derive their axiomatization, so in the case where **TC** and **C** coincide, our axiomatization is the same as theirs.

Proposition (Parameterized dinaturality) (for $\mathsf{f} : Z \otimes U(B) \to U(A)$ and $\mathsf{g} : X \otimes U(A) \to Y \otimes U(B)$ in **C**):

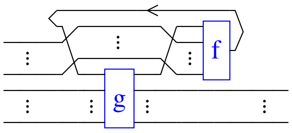


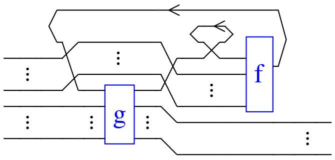
=

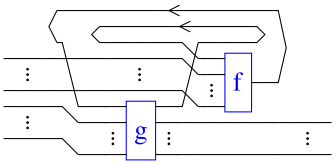


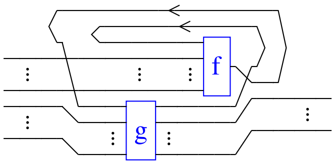
$$\mathrm{Tr}_{(X \otimes Z)\,Y\,A}\,((\mathsf{id}_X \otimes \mathsf{symm}_{Z\,U(A)}); (\mathsf{g} \otimes \mathsf{id}_Z); (\mathsf{id}_Y \otimes (\mathsf{symm}_{U(B)\,Z}; \mathsf{f}))) = \mathrm{Tr}_{(X \otimes Z)\,Y\,B}\,((\mathsf{id}_X \otimes \mathsf{f}); \mathsf{g})$$

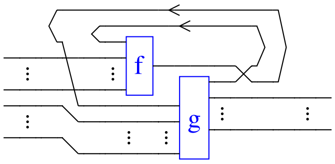
Proof.

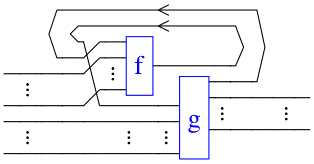


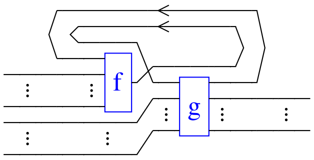
=  (Yanking)

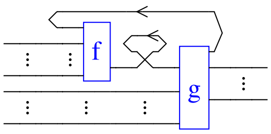
=  (Naturality of Tr)

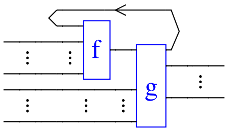
=  (Naturality of symm)

=  (Functoriality of $\otimes$)

=  (Symmetry sliding)

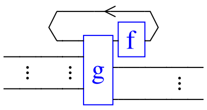
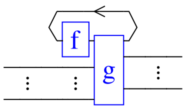
=  (Naturality of symm)

=  (Naturality of Tr)

=  (Yanking)

□

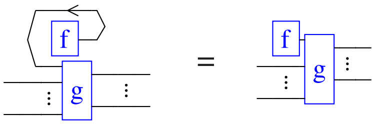
Corollary (Dinaturality) (for f : $U(B) \to U(A)$ and g : $X \otimes U(A) \to Y \otimes U(B)$ in **C**):

 = 

$$\mathrm{Tr}_{X\,Y\,A}\,(\mathrm{g};\,(\mathrm{id}_Y \otimes \mathrm{f})) \;=\; \mathrm{Tr}_{X\,Y\,B}\,((\mathrm{id}_X \otimes \mathrm{f});\,\mathrm{g})$$

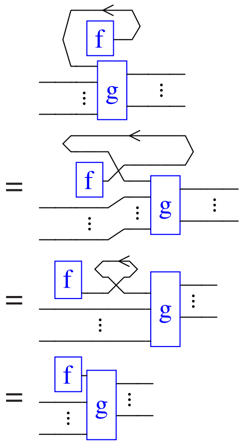
Proof. Let Z be I in the above. □

Proposition (Trace respects unit) (for $f : I \to U(A)$ and $g : X \otimes U(A) \to Y$ in **C**):



$$\mathrm{Tr}_{X\,Y\,A}\,(g; \mathrm{id}_Y \otimes f) = (\mathrm{id}_X \otimes f); g$$

Proof.



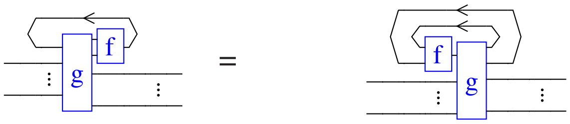
(Acyclic graph iso)

(Naturality of Tr)

(Yanking)

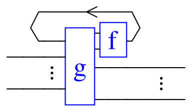
□

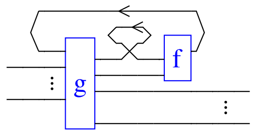
Proposition (Trace respects tensor) (for $f : U(B) \otimes U(C) \to U(A)$ and $g : X \otimes U(A) \to Y \otimes U(B) \otimes U(C)$ in **C**):

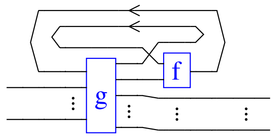


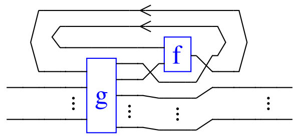
$$\mathrm{Tr}_{X\,Y\,A}\,(g; \mathrm{id}_Y \otimes f) = \mathrm{Tr}_{X\,Y\,C}\,(\mathrm{Tr}_{X\,Y\,B}\,((\mathrm{id}_X \otimes f); g))$$

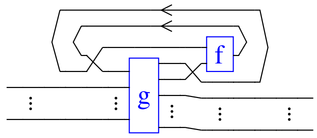
Proof.

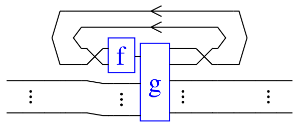


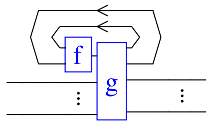
=  (Yanking)

=  (Naturality of Tr)

=  (Acyclic graph iso)

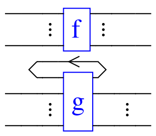
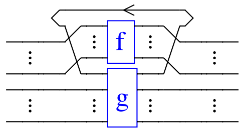
=  (Symmetry sliding)

=  (Parameterized dinaturality)

=  (Symmetry sliding)

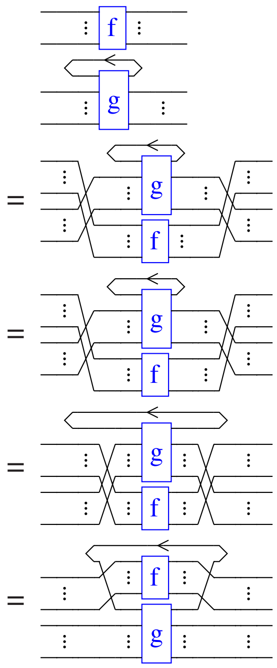
□

Proposition (Trace respects double symmetry) (for f : $V \to W$ and g : $X \otimes U(A) \to Y \otimes U(A)$ in **C**):

 = 

$\mathrm{Tr}_{X\,Y\,A}\,(\mathrm{g}) \otimes \mathrm{f} \;=\; \mathrm{Tr}_{(X \otimes V)\,(Y \otimes W)\,A}\,((\mathrm{id}_X \otimes \mathrm{symm}_{V\,U(A)});\,(\mathrm{g} \otimes \mathrm{f});\,(\mathrm{id}_Y \otimes \mathrm{symm}_{U(A)\,W}))$

Proof.



= (Acyclic graph iso)

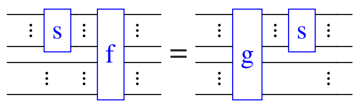
= (Strength of Tr)

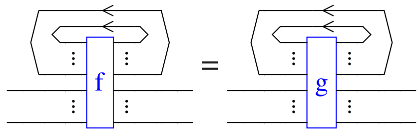
= (Naturality of Tr)

= (Acyclic graph iso)

□

We show below that the axioms for a partially traced monoidal category are complete for graphs up to graph isomorphism. But as we discussed before, graph isomorphism is not a sound model for categories with finite products, and we chose bisimulation as the appropriate equivalence on graphs. In order to provide a complete axiomatization for cyclic graphs up to bisimulation, we need an additional property. In a partially traced monoidal category where the monoid is product, the trace is *uniform wrt shuffles* iff, for any shuffle s:

If: 

then: 

A *partially traced cartesian category* is a partially traced monoidal category where the monoid is product and the trace is uniform wrt shuffles. We shall show below that the axioms for a partially traced cartesian category are sound and complete for traceable cyclic graphs up to bisimulation.

We can now replay the same proof of soundness and completeness as in the finite products case. In some cases, the proof of the theorems is so similar to that in the finite products case that we elide it.

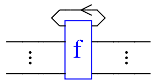
Let CGraph(Σ_V) be the category of traceable cyclic single-coloured flow graphs over Σ_V viewed up to bisimulation.

Proposition (Bisimulation is a congruence). *Bisimulation is a congruence wrt the graph operations.*

Proposition (Flow graphs form a partially traced cartesian category). *CGraph(Σ_V) is a partially traced cartesian category.*

Define CTerm(Σ_V) as the free partially traced cartesian category with finite products over Σ_V.

f,g ::= ...as for Term...

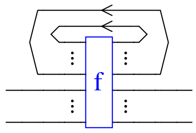
| Tr_X(f) 

If terms can be proved equal using the axioms for a partially traced cartesian category, we shall write $\vdash$ f = g.

Proposition (Soundness). *If $\vdash f = g$ then $[[f]] \sim [[g]]$.*

Proposition (Expressivity). *For any traceable graph G, there is a term f such that $[[f]] \sim G$.*

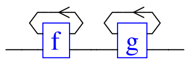
A term is in *normal form* (ranged over by n) iff it is of the form:



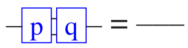
where f is acyclic, and the only traceable edges in f are the incoming and outgoing edges.

Proposition (Normalization). *For any term f there is a normal form g such that $\vdash f = g$.*

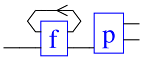
Proof. An induction on f. The only difficult case is composition. By induction we have (drawing the one-cycle case for simplicity: the multiple-cycle case is dealt with the same way):



We can find permutations p and q such that:

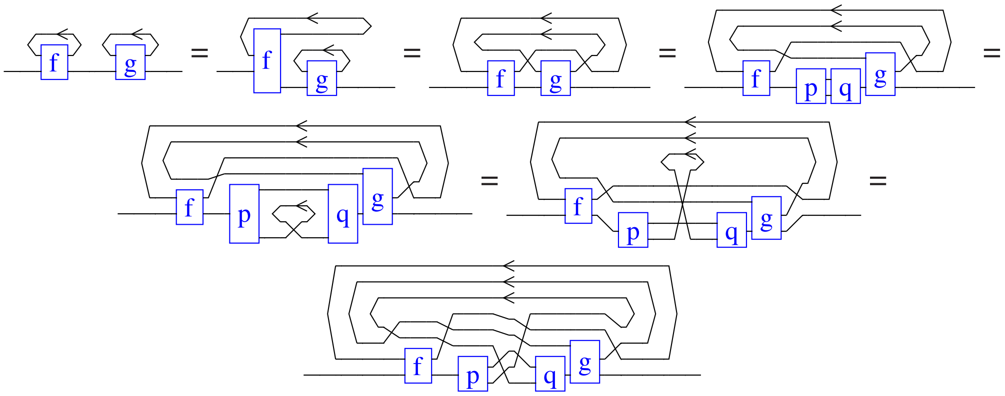


and:



: **X** $\rightarrow$ **Y**,**Z** (**Y** a vector of traceable sorts)

Again, we shall only prove the case where **Y** is a singleton vector, the more general case follows in the same way.



So we have normalized the composition. The other cases are simpler. □

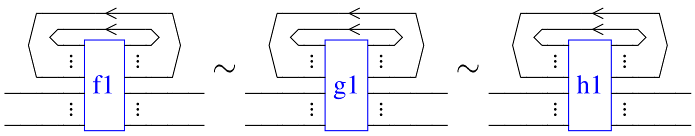
We can then prove completeness, essentially by converting any bisimulation between cyclic graphs into a shuffle, then using uniformity.

Proposition (Completeness). *If [[f]] ∼ [[g]] then* $\vdash f = g$.

Proof. Let R be the bisimulation between [[f]] and [[g]]. We can regard this as a graph G:

- Nodes are pairs $(N,N') \in R$.
- Edges are pairs $(E,E') \in R$.
- Incoming edges, outgoing edges, and labellings inherited from [[f]] and [[g]] (since R is a graph relation, this is consistent).

By expressivity, let h be the term such that [[h]] $\sim G$. Find normal forms for f, g and h:

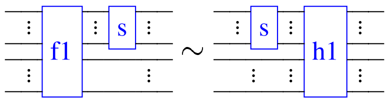


Let $E_1 ... E_m$ be the outgoing edges of [[f1]] which form the cycles in [[f]], and $F_1 ... F_n$ be the outgoing edges of [[h1]] which form the cycles in [[h]]. Construct a shuffle s with:

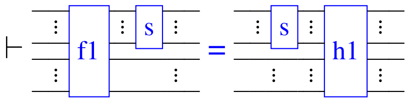
- Edges $E_1 ... E_m$.
- Incoming edges $E_1 ... E_m$.

- Outgoing edges $E'_1 ... E'_n$, where $E'_i = E_j$ whenever $F_i = (E_j, \_)$.
- Labelling inherited from [[f1]].

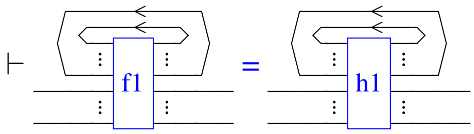
We can then show that:



and since we have completeness for acyclic graphs, this means:



Then since trace is uniform wrt shuffles:



and so $\vdash$ f = h. We can show $\vdash$ g = h similarly.

Proposition (Initiality). *CGraph*(Σ_V) *is the initial partially traced cartesian category over* Σ_V.

We can extend this result to include embedding single-coloured cyclic graphs into two- or three-coloured graphs. Let CGraph(Σ_V,Σ_C) and CGraph(Σ_V,Σ_V,Σ_P) be the categories of cyclic two- and three-coloured graphs such that:

- Any cyclic path goes through at least one edge labelled with a **traceable** sort.
- Any nodes on a cycle are labelled in Σ_V.

Proposition (Initiality). *CGraph*(Σ_V,Σ_C) *is the initial symmetric monoidal category over* Σ_C *with Graph*(Σ_V) *as a sub smc with the same objects. CGraph*(Σ_V,Σ_C,Σ_P) *is the initial symmetric premonoidal category over* Σ_P *with CGraph*(Σ_V,Σ_C) *as centre.*

E Premonoidal categories and state transformers

The graphical presentation of symmetric premonoidal categories uses extra control arcs to model control dependencies. This can be seen as a simple form of *state transformer* where each graph has one extra incoming arc and one extra outgoing arc.

Given a strict symmetric monoidal category C with a chosen object S, the *state transformer category* of C, State(C) is given by:

- Objects are those of C.
- The homset $\mathsf{State}(\mathsf{C})[X, Y]$ is $\mathsf{C}[X \otimes \mathsf{S}, Y \otimes \mathsf{S}]$.

The identity-on-objects inclusion $\mathsf{C} \hookrightarrow \mathsf{State}(\mathsf{C})$ maps f to $\mathsf{f} \otimes \mathsf{id}_{\mathsf{S}}$.

$\mathsf{C} \hookrightarrow \mathsf{State}(\mathsf{C})$ is a strict symmetric premonoidal category with premonoidal structure given by:

- $\mathsf{f} \oslash \mathsf{g}$ is $\mathsf{f} \otimes \mathsf{g}$.
- $\mathsf{f} \obslash \mathsf{g}$ is $(\mathsf{id} \otimes \mathsf{symm})$; $(\mathsf{f} \otimes \mathsf{g})$; $(\mathsf{id} \otimes \mathsf{symm})$.

The state transformer construction is well-known, and is given by Power and Robinson as one of the canonical examples of a symmetric premonoidal category. We shall now show that state transformers are canonical strict symmetric premonoidal categories by showing that for *any* strict symmetric premonoidal category $\mathsf{C} \hookrightarrow \mathsf{P}$ we can find a strict symmetric monoidal category D such that:

$$\begin{tikzcd} \mathsf{C} \arrow[r, hook] \arrow[d, hook, "\mathrm{full}"] & \mathsf{P} \arrow[d, hook, "\mathrm{full}"] \\ \mathsf{D} \arrow[r, hook] & \mathsf{State}(\mathsf{D}) \end{tikzcd}$$

that is $\mathsf{C} \hookrightarrow \mathsf{P}$ is a full sub-strict-symmetric-premonoidal-category of $\mathsf{D} \hookrightarrow \mathsf{State}(\mathsf{D})$.

The rest of this section shows how to construct D, and show that it has the required properties.

Let $\mathsf{LGraph}(\Sigma_{\mathsf{C}})$ be the subcategory of $\mathsf{Graph}(\Sigma_{\mathsf{V}},\Sigma_{\mathsf{C}})$ category of *linear* flow graphs, that is ones where:

- All nodes are labelled from Σ_{C}.
- Each edge occurs exactly once either as an outgoing edge of the graph, or an incoming edge of a node.

Note in particular that since we cannot duplicate or discard edges, this means this category has no diagonal or terminal. In fact, we can adapt the previous proof to show that this is the initial strict symmetric monoidal category over Σ_{C}.

Similarly, let $\mathsf{LGraph}(\Sigma_{\mathsf{C}},\Sigma_{\mathsf{P}})$ be the subcategory of $\mathsf{Graph}(\Sigma_{\mathsf{V}},\Sigma_{\mathsf{C}},\Sigma_{\mathsf{P}})$ where:

- All nodes are labelled from Σ_{C} or Σ_{P}.
- Each edge occurs exactly once either as an outgoing edge of the graph, or an incoming edge of a node.

We can adapt the previous proof to show that this is the initial strict symmetric premonoidal category over Σ_P with centre $\mathrm{LGraph}(\Sigma_C)$.

Given two signatures Σ_C and Σ_P with the same sorts, define $\mathrm{Mix}(\Sigma_C,\Sigma_P)$ to be the signature with:

- Sorts those of Σ_C plus a new sort S.
- Constructors the disjoint union of those of Σ_C and those of Σ_P.
- $\mathrm{c} : \mathbf{X} \to \mathbf{Y}$ in $\mathrm{Mix}(\Sigma_C,\Sigma_P)$ whenever $\mathrm{c} : \mathbf{X} \to \mathbf{Y}$ in Σ_C.
- $\mathrm{c} : \mathbf{X}\mathrm{S} \to \mathbf{Y}\mathrm{S}$ in $\mathrm{Mix}(\Sigma_C,\Sigma_P)$ whenever $\mathrm{c} : \mathbf{X} \to \mathbf{Y}$ in Σ_P.

We then have:

$$\begin{tikzcd}
\mathrm{LGraph}(\Sigma_C) \arrow[r, hook] \arrow[d, hook, "\text{full}"] & \mathrm{LGraph}(\Sigma_C,\Sigma_P) \arrow[d, hook, "\text{full}"] \\
\mathrm{LGraph}(\mathrm{Mix}(\Sigma_C,\Sigma_P)) \arrow[r, hook] & \mathrm{State}(\mathrm{LGraph}(\mathrm{Mix}(\Sigma_C,\Sigma_P)))
\end{tikzcd}$$

where the full subcategories are taken by only including objects which do not include the new sort S.

We can regard a monoidal category C as a signature with:

- Sorts are objects.
- Constructors $f : X_1,\ldots,X_m \to Y_1,\ldots,Y_n$ are morphisms $f : X_1 \otimes \ldots \otimes X_m \to Y_1 \otimes \ldots \otimes Y_n$.

and similarly for premonoidal categories. Any category is trivially a category over itself, and so we have a unique symmetric premonoidal functor:

$$\begin{array}{rl}
[\![_]\!]_C : & \mathrm{LGraph}(\mathrm{C}) \twoheadrightarrow \mathrm{C} \\
[\![_]\!]_P : & \mathrm{LGraph}(\mathrm{C},\mathrm{P}) \twoheadrightarrow \mathrm{P}
\end{array}$$

such that:

$$[\![-\boxed{\mathrm{c}}-]\!]_C = c \qquad [\![=\boxed{\mathrm{d}}=]\!]_P = d$$

and so we can define $\mathrm{D} \hookrightarrow \mathrm{Q}$ as a pushout in the category of strict symmetric premonoidal categories:

$$
\begin{array}{rrcl}
[[\_]]_{\mathrm{C}}: & \mathrm{LGraph}(\mathrm{C}) & \twoheadrightarrow & \mathrm{C} \\
& \downarrow & & \downarrow \\
[[\_]]_{\mathrm{D}}: & \mathrm{LGraph}(\mathrm{Mix}(\mathrm{C},\mathrm{P})) & \twoheadrightarrow & \mathrm{D}
\end{array}
$$

$$
\begin{array}{rrcl}
[[\_]]_{\mathrm{P}}: & \mathrm{LGraph}(\mathrm{C},\mathrm{P}) & \twoheadrightarrow & \mathrm{P} \\
& \downarrow & & \downarrow \\
[[\_]]_{\mathrm{Q}}: & \mathrm{State}(\mathrm{LGraph}(\mathrm{Mix}(\mathrm{C},\mathrm{P}))) & \twoheadrightarrow & \mathrm{Q}
\end{array}
$$

Proposition (Full sub-strict-symmetric-premonoidal-category). *$C \hookrightarrow P$ is a full sub-strict-symmetric-premonoidal category of $D \hookrightarrow Q$.*

Proof. Throughout this proof, we shall give the reasoning for the premonoidal category, since the reasoning for the central category is identical.

We have a functor $F : \mathrm{P} \to \mathrm{Q}$ given by the pushout diagram. We will first show that this functor is monic.

For any strict symmetric premonoidal category P let $\mathrm{P}_\perp$ be the strict symmetric premonoidal category given by adjoining a new object $\perp$ and making the tensors strict. Let lift be the functor $\mathrm{P} \hookrightarrow \mathrm{P}_\perp$.

We can find a strict symmetric premonoidal functor test such that:

$$
\begin{array}{rrcl}
\mathrm{lift}: & \mathrm{LGraph}(\mathrm{C},\mathrm{P}) & \to & \mathrm{LGraph}(\mathrm{C},\mathrm{P})_\perp \\
& \downarrow & & \| \\
\mathrm{test}: & \mathrm{State}(\mathrm{LGraph}(\mathrm{Mix}(\mathrm{C},\mathrm{P}))) & \to & \mathrm{LGraph}(\mathrm{C},\mathrm{P})_\perp
\end{array}
$$

such that $\mathrm{test}(\mathrm{S}) = \perp$.

Then:

$$
\begin{array}{ccccc}
\mathrm{LGraph}(\mathrm{C},\mathrm{P}) & & \twoheadrightarrow & & \mathrm{P} \\
\downarrow & & & & \downarrow \\
\mathrm{State}(\mathrm{LGraph}(\mathrm{Mix}(\mathrm{C},\mathrm{P}))) & \to & \mathrm{LGraph}(\mathrm{C},\mathrm{P})_\perp & \twoheadrightarrow & \mathrm{P}_\perp
\end{array}
$$

and since we defined Q as a pushout, this means we have a unique symmetric premonoidal functor $\mathrm{test}' : \mathrm{Q} \to \mathrm{P}_\perp$ making the pushout diagram commute. In particular, we have $F \mathbin{;} \mathrm{test}' = \mathrm{lift}$, and since lift is monic, so is F.

For fullness, for any $\mathrm{f} : F(X) \to F(Y)$ in Q we can find (since $[[\_]]_{\mathrm{Q}}$ is epi) a graph $G : \mathbf{X} \to \mathbf{Y}$ such that:

$$
[[G : \mathbf{X} \to \mathbf{Y}]]_{\mathrm{Q}} = \mathrm{f} : F(X) \to F(Y)
$$

Since $\mathrm{test}'(F(X)) \neq \perp$ we have $\mathrm{S} \notin \mathbf{X}$, so $\mathbf{X}$ is in $\mathrm{LGraph}(\mathrm{C},\mathrm{P})$, and similarly for $\mathbf{Y}$. Since $\mathrm{LGraph}(\mathrm{C},\mathrm{P}) \hookrightarrow \mathrm{State}(\mathrm{LGraph}(\mathrm{Mix}(\mathrm{C},\mathrm{P})))$ is a full subcategory, G is in $\mathrm{LGraph}(\mathrm{C},\mathrm{P})$

and so $[[G]]_{\mathrm{P}}$ is in P. By the pushout diagram, $F[[G]]_{\mathrm{P}} = \mathrm{f}$, and so P is a full sub-strict-symmetric-premonoidal-category of Q. □

Proposition (Q is State(D)). *Q is isomorphic to State(D).*

Proof. Since the pushout defining Q was taken in the category of strict symmetric premonoidal categories, the following diagram commutes:

$$\begin{tikzcd} {[[\_]]_{\mathrm{D}}:} & \mathrm{LGraph}(\mathrm{Mix}(\mathrm{C},\mathrm{P})) \arrow[r, two heads] \arrow[d, hook] & \mathrm{D} \arrow[d, hook] \\ {[[\_]]_{\mathrm{Q}}:} & \mathrm{State}(\mathrm{LGraph}(\mathrm{Mix}(\mathrm{C},\mathrm{P}))) \arrow[r, two heads] & \mathrm{Q} \end{tikzcd}$$

We will now define an identity-on-objects isomorphism between Q and State(D):

- For any $\mathrm{f} : X \to Y$ in Q we can find (since $[[\_]]_{\mathrm{Q}}$ is epi) a graph $G : \mathbf{X} \to \mathbf{Y}$ in State(LGraph(Mix(C,P))) such that:

  $$[[G : \mathbf{X} \to \mathbf{Y}]]_{\mathrm{Q}} = \mathrm{f} : X \to Y$$

  By definition, $G : \mathbf{X}\mathrm{S} \to \mathbf{Y}\mathrm{S}$ in LGraph(Mix(C,P)) and so define:

  $$\mathrm{st}(\mathrm{f}) = [[G : \mathbf{X}\mathrm{S} \to \mathbf{Y}\mathrm{S}]]_{\mathrm{D}} : X \otimes \mathrm{S} \to Y \otimes \mathrm{S} \text{ in } \mathrm{D}$$

  which gives us:

  $$\mathrm{st}(\mathrm{f}) : X \to Y \text{ in } \mathrm{State}(\mathrm{D})$$

- For any $\mathrm{f} : X \to Y$ in State(D), we have $\mathrm{f} : X \otimes \mathrm{S} \to Y \otimes \mathrm{S}$ in D. We can find (since $[[\_]]_{\mathrm{D}}$ is epi) a graph $G : \mathbf{X}\mathrm{S} \to \mathbf{Y}\mathrm{S}$ in LGraph(Mix(C,P)) such that:

  $$[[G : \mathbf{X}\mathrm{S} \to \mathbf{Y}\mathrm{S}]]_{\mathrm{D}} = \mathrm{f} : X \otimes \mathrm{S} \to Y \otimes \mathrm{S} \text{ in } \mathrm{D}$$

  By definition, $G : \mathbf{X} \to \mathbf{Y}$ in State(LGraph(Mix(C,P))) and so define:

  $$\mathrm{ts}(\mathrm{f}) = [[G : \mathbf{X} \to \mathbf{Y}]]_{\mathrm{Q}} : X \to Y \text{ in } \mathrm{Q}$$

It is routine to show that st and ts are symmetric premonoidal functors, and that they form an isomorphism. □

Thus for any strict symmetric premonoidal category $\mathrm{C} \hookrightarrow \mathrm{P}$ we can find an strict symmetric monoidal category D such that:

$$\begin{tikzcd} \mathrm{C} \arrow[r, hook] \arrow[d, hook, "\mathrm{full}"] & \mathrm{P} \arrow[d, hook, "\mathrm{full}"] \\ \mathrm{D} \arrow[r, hook] & \mathrm{State}(\mathrm{D}) \end{tikzcd}$$

F Bibliography

Benton, P.N. A mixed linear and non-linear logic: proofs, terms and models, *Proc. Computer Science Logic 1995*, Springer LNCS 933 and Univ. Cambridge Technical Report 352, 1994..

Gardner, P.A., A name-free account of action calculi, in *Proc. MFPS '95*, Electronic Notes in Comp. Sci. 1, Elsevier, 1995.

Hasegawa, M. *Models of Sharing Graphs (A Categorical Semantics of Let and Letrec)*, Ph.D thesis, Univ. Edinburgh, LFCS report ECS-LFCS-97-360, 1997.

Hasegawa, M. Recursion from cyclic sharing: traced monoidal categories and models of cyclic lambda calculi, in *Proc. 3rd International Conference on Typed Lambda Calculi and Applications* (*TLCA'97*), Springer LNCS 1210, pp. 196-213, 1997.

Joyal, A., Street, R.H. and Verity, D., Traced monoidal categories, in *Math. Proc. Cambridge Philosophical Soc.* 119(3), pp. 425-446, 1996.

Milner, R., A complete inference system for a class of regular behaviours, *J. Comput. System Sci.* 28, pp. 439-466, 1984.

Milner, R., Calculi for interaction, *Acta Informatica* 33(8), pp. 707-737, 1996.

de Paiva, V.C.V. and Ritter, E. On explicit substitutions and names, extended abstract in *Proc. ICALP '97*, Springer LNCS 1104, pp. 17-31, full version as Univ. Birmingham technical report CSRP-97-5, 1997.

Power, A.J. Premonoidal categories as categories with algebraic structure, to appear in *Proc. MFPS '96*.

Power, A.J. and Robinson, E.P. *Premonoidal categories and notions of computation*, to appear in *Math. Struct. in Comput. Science*, also available electronically.

Power, A.J. and Thielecke, H. Environments, continuation semantics and indexed categories, in *Proc. TACS '97*, Springer LNCS 1281, pp. 391-414, 1997.

Selinger, P. *Control Categories: an Axiomatic Approach to the Semantics of Control in Functional Languages*, Manuscript