Constructing fully complete models for multiplicative linear logic

DOI:
10.1109/LICS.2012.67

Citation for published version (APA):

Published in:

Citing this paper
Please note that where the full-text provided on Manchester Research Explorer is the Author Accepted Manuscript or Proof version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version.

General rights
Copyright and moral rights for the publications made accessible in the Research Explorer are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Takedown policy
If you believe that this document breaches copyright please refer to the University of Manchester’s Takedown Procedures [http://man.ac.uk/04Y6Bo] or contact uml.scholarlycommunications@manchester.ac.uk providing relevant details, so we can investigate your claim.
Constructing Fully Complete Models for Multiplicative Linear Logic

Andrea Schalk and Hugh Steele
School of Computer Science
The University of Manchester
M13 9PL, United Kingdom
Email: {firstname.lastname}@cs.man.ac.uk

Abstract—We demonstrate how the Hyland-Tan double glueing construction produces a fully complete model of the unit-free multiplicative fragment of Linear Logic when applied to any of a large family of degenerative ones. This process explains as special cases a number of such models which appear in the literature. In order to achieve this result, we make use of a tensor calculus for compact closed categories with finite biproducts. We show how the combinatorial properties required for a fully complete model are obtained by the construction adding to those already available from the original category.

Index Terms—Linear Logic; Full Completeness; Double Glueing; Compact Closure

I. INTRODUCTION

Linear Logic [7] is a well-known formal system that has attracted interest from computer science as well as logicians. It has a very well behaved proof theory, and categorical models for linear logic also contain a model of the simply-typed \(\lambda\)-calculus. Fully complete [1] models are those that equate precisely those proofs considered equivalent by the proof theory, and which exclusively contain morphisms that are the interpretation of some proof.

The best studied fragment of Linear Logic is that of unit-free Multiplicative Linear Logic, MLL\(^{-}\). To model this one requires a *-autonomous category [2], but not all such categories satisfy the desired full completeness property. Studies of ‘good’ models in this stronger sense are [1], [9], [11], [12], [6]. Degenerate models for linear logic are given by compact closed categories.

In [14] the double glueing construction is introduced (see also [10]), and it is suggested that when this is applied to three particular compact closed categories then fully complete models may be obtained. The three categories in question are the category Rel of sets and relations, the category FVec\(_F\) of finite dimensional vector spaces over an arbitrary field \(F\) of characteristic 0, and the category of Conway games and history-free strategies. However, the proof of the second—arguably the most interesting case—is not completed in the cited work; and the restriction on the characteristic of the field turns out to be unnecessary. Furthermore no two of the three proofs lend themselves to a common unification.

In this paper we show how to complete this proof and, moreover, how to apply the same methodology to a large variety of models provided by tensor-generated compact closed categories with biproducts. In the process we illustrate how a ‘tensor calculus’ can be applied when reasoning about such categories, and how that has certain combinatorial consequences.

We proceed by briefly describing the proof theory of MLL\(^{-}\) and how this system can be modelled categorically. We formally introduce notions of full completeness for such models, and indicate what can be said about the models provided by compact closed categories. An introduction to double glueing follows, and we then give an account of our ‘tensor calculus’ for the resulting categories. We conclude with a sketch of a proof for a strong full completeness theorem. Our main result, namely that all tensor-generated compact closed categories with finite biproducts are fully complete MLL\(^{-}\) models, comes about as a corollary of this theorem.

II. MLL\(^{-}\) PROOF NETS

MLL\(^{-}\) possesses a beautiful proof theory revolving around the notion of proof nets. Proof nets provide a method of equating two distinct derivations which differ only due to ‘bureaucracy’ [7]. Since derivations in MLL\(^{-}\)can be normalised confluently, we are interested primarily in cut-free MLL\(^{-}\) proof nets. It is sufficient to consider right-sided sequents of formulae built from literals (which in a derivation are created as pairs, one positive, one negative) using the multiplicative conjunction \(\otimes\) and disjunction \(\oplus\). Given a deduction in the system we can construct a graph by using the parse trees of the final sequent, connecting those literals that are created together in said proof via edges known as \(\text{axiom links}\). This is the proof net that corresponds to the derivation, and we wish to equate those deductions that have the same proof net.

While every proof net represents a derivation, it is possible to create graphs which resemble proof nets but do not correspond to correct derivations. Given a parse tree for a sequent constructed from literals and the MLL\(^{-}\) connectives \(\otimes\) and \(\oplus\) we use the term proof structure for a graph resulting from connecting matching literals.

It is possible to check whether a given proof structure is, in fact, a proof net [5] using certain correctness criteria. A switching of a proof structure is a subgraph created by removing exactly one of the two argument edges of each \(\oplus\)-vertex.

A proof structure is a proof net if and only if every one of its switchings is both acyclic and connected.
A. MDNF Proof Nets

We say that an MLL\(^{-}\) formula \(F\) is in *multiplicative disjunctive normal form* or is an MDNF formula provided that 
\[
F = \bigotimes_{m=1}^{M} \bigotimes_{i=1}^{I(m,j)} F(m,j)
\]
for some atoms \(L_{11}, \ldots, L_{M_{R}}\), and some function \(\phi\) to describe whether the literal to which it is connected is positive or negative; and that a sequent or proof structure is MDNF if each of its formulae is in the prescribed form.

Formulae and sequents in this form can be thought of as blocks of literals in tensor products possibly joined together by \(\otimes\) connectives. We refer to these tensor products simply as *blocks* from now on.

Proof structures for a multiplicative disjunctive normal form are easily checked for correctness. If such a proof structure has a switching which contains a cycle, then that cycle cannot contain a \(\otimes\)-vertex. Similarly, two blocks are connected in a switching of a given proof structure if and only if they can be connected through a path of axiom links between them. As such, since each \(\otimes\)- and \(\otimes\)-vertex must be connected to at least one literal in any switching, we know that the proof structure itself is connected if and only if every block is connected to every other block by a path of axiom links. Therefore only blocks (which are subtrees of a structure) and the links between them are important in deciding the correctness of a proof structure, so we can use a simplified version of the correctness criteria.

Proposition 1. An MDNF proof structure is a proof net if and only if its maximal \(\otimes\)-free subgraph is a tree.

III. CATEGORICAL MODELS

Sound categorical models of MLL\(^{-}\) are found in the form of **autonomous categories** [13]—these are symmetric monoidal categories with a well-behaved self-duality. The underlying idea is quite simple: Each symbol in the logic is interpreted by a functor (of the appropriate arity) on the category; the monoidal structure \(\otimes\) gives conjunction, the duality \((-)^{\dagger}\) allows negation, and to model disjunction these can be combined to define a De Morgan dual \(\Psi = ((-)^{\dagger} \otimes (-)^{\dagger})^{\dagger}\). We use the latter functor also to interpret the commas separating formulae in a sequent.

Hence every sequent in MLL\(^{-}\) determines a functor

\[
\mathbb{C}^N \times (\mathbb{C}^{op})^N \rightarrow \mathbb{C}
\]

where \(N \in \mathbb{N}^+\). If we look at these functors then we see that they are built by

- applying the duality functor \((-)^{\dagger}\) to each copy of \(\mathbb{C}^{op}\),
- creating as many copies of the arguments as required, then reordering them appropriately,
- applying the functors \(\otimes\) and \(\Psi\) to get a result in \(\mathbb{C}\).

We refer to the functors that can be built in this way as **MLL\(^{-}\) functors**. Similarly, functors that correspond to MDNF sequents are referred to as **MDNF functors**.

1From here on functions named \(\phi\) or \(\Psi\) take the same form as the \(\phi\) given here and are used for the same purpose.

Assume we have a right-sided sequent interpreted by the MLL\(^{-}\) functor \(F\). The formula representing truth is interpreted by the functor whose value is the constant \(I\), the unit for the monoidal structure. We refer to this functor as \(\mathcal{K}\), and allow ourselves to adjust its source as needed. Every proof of the given sequent is interpreted by a dinatural transformation from \(\mathcal{K}\) (with the same source as \(F\)) to \(F\), which is a family of arrows \((r_r \in \mathbb{C}[I,F(R,R)])_{r \in \mathbb{C}^{N}}\) for which the diagram below commutes for every choice of \(f = (f_1, \ldots, f_N): R \rightarrow S\), where \(R, S \in \mathbb{C}^N\).

![Diagram](image)

We use the term **MLL\(^{-}\) (MDNF) transformation** to refer to any dinatural transformation from \(\mathcal{K}\) to an MLL\(^{-}\) (MDNF) functor.

It should be noted that dinatural transformations, unlike natural transformations, cannot be guaranteed to compose with one another. However, it is always possible to compose them with natural transformations to produce other dinatural transformations. Given a natural transformation \(\mu\) between \(F\) and another MLL\(^{-}\) functor \(G\), we obtain a new MLL\(^{-}\) transformation \(\mu \circ \tau = (\mu_{R,R} \circ \tau_R \in \mathbb{C}[I,G(R,R)])_{R \in \mathbb{C}^{N}}\). The outside of the diagram underneath commutes for all \(R\) and \(S\) due to the (di)naturality of the collections of arrows being used.

One can now show that two proofs of the same sequent are interpreted by the same dinatural transformation if and only if they have the same proof net. In other words, this categorical interpretation of proofs fits very well with the existing proof theory for MLL\(^{-}\).

A. Full Completeness

Full Completeness was first defined in [1], and it is meant to describe the tightest possible connection between the logic and its model. Not only are the interpretations of two proofs the same if and only if they are equivalent in the proof theory, but the model does not contain any ‘non-proofs’.

572
Definition 2. A \(*\)-autonomous category \(C\) satisfies MLL− (MDNF) full completeness if every MLL− (MDNF) transformation in the category is the interpretation of a cut-free proof net.

It is desirable, but far from easy, to find fully complete models. We separate our proof into showing MDNF full completeness before establishing that the nominally stronger concept of MLL− full completeness is a consequence.

IV. COMPACT CLOSED CATEGORIES AS MODELS

Compact closed categories are particular degenerate \(*\)-autonomous ones: the \(\otimes\) and \(\exists\) functors for these categories are naturally isomorphic. Nonetheless it turns out that they can form the basis for constructing good models.

There is a lot of structure here: in particular for every compact closed category \(C\) the set of endomorphisms on the tensor unit \(I\) is a commutative monoid under composition, and \(C\) is enriched over the corresponding category of actions. For this reason \(C[I,I]\) is also known as the set of scalars of the category.

We are interested in the case where there is even more structure, namely where the compact closed category \(C\) has finite biproducts. Recall [8] that \(C\) has finite products if and only if it has finite sums if and only if it has finite biproducts. In this case the scalars form a commutative unital semiring, and due to the biproducts distributing over the tensor product \(\otimes\) is enriched over 

\[\text{CSMod}_{C[I,I]}\]—the category of commutative semimodules over the semiring of scalars. Note that such categories also have zero morphisms between any two objects.

It is easy to show that in such a category every linear combination of dinatural transformations is another such, giving another obstacle to full completeness. Nonetheless there is a meaningful concept of such a category being ‘as fully complete as it can hope to be’.

Definition 3. A compact closed category \(C\) with finite biproducts satisfies feeble full completeness if every MLL− transformation for the category is a linear combination of interpretations of proof structures over the same sequent.

It easily follows from [3] that finite biproductual compact closed categories whose tensor unit acts as a generator satisfy feeble full completeness. These models are called tensor-generated, and there are many of them. Examples include both \(\text{Rel}\) and \(\text{FDVec}\) for any field \(F\), and extend beyond these to include categories such as that of finite-dimensional Hilbert spaces. Every compact closed category with finite biproducts clearly has a non-trivial full subcategory which is tensor-generated, namely the category generated by the object \(I\) and the tensor, biproduct and duality functors.

V. DOUBLE GLUEING

The double gluing construction was originally described in [14] and has since been generalised [10]. It can be thought of as adding structure to objects, which then has to be preserved by morphisms. As a result, there are fewer arrows in the resulting homsets. We define \(\perp = I^\perp\).

Given a \(*\)-autonomous category \((C, \otimes, \iota, (-)^\perp)\) we define the double-joined \(*\)-autonomous category \(GC\):

\(\text{Obj}: (R, U, X) \text{ with } R \in \text{C}, U \subseteq C[I,R], X \subseteq C[R, \perp].\)

\(\text{Morph}: f: R \rightarrow S \text{ is an arrow from } (R, U, X) \text{ to } (S, V, Y).\)

The structure, up to canonical isomorphism, is given by

1. \(I = (I, \{1\}, C[I, \perp])\),
2. \((R, U, X) \otimes (S, V, Y) = (R \otimes S, U \otimes V, Z)\), where \(Z\) is
   \[\{A \otimes B \rightarrow \perp : \forall v \in V, \langle v; z\rangle_B \in X, \forall u \in U \langle u; z\rangle_L \in Y\},\]
3. \((R, U, X)^\perp = (R^\perp, X^\perp, U^\perp)\)

where

\[\langle v; z\rangle_R = R \triangleright R \otimes I \xrightarrow{1_R \otimes v} R \otimes S \xrightarrow{\pi} \perp\]

and similarly for \(\langle u; z\rangle_L\).

In general, when discussing a \(GC\)-object \((R, U, X)\), we refer to \(U\) and \(X\) as the object’s sets of values and covalues respectively, and write \((R, U, X)_{\text{Val}} = U\) and \((R, U, X)_{\text{CoVal}} = X\). Note that the set of arrows in \(GC\) from \(I\) to \((R, U, X)\) is equal to \(U\). Also note that there is a forgetful functor \([-\] : \(GC \rightarrow C\) which maps an object \((R, U, X)\) to the underlying \(C\) object \(R\). It can be shown that every MLL− transformation \(\tau\) to a functor \(F\) in \(GC\) is uniquely determined by an underlying MLL− transformation to \([F]\) in \(C\).

VI. MULTILINEAR ALGEBRA

From now on we assume that we are concerned with a compact closed category with biproducts \(C\). Such categories, by virtue of their enrichment over a category of commutative semimodules, allow us to use linear algebra to carry out a number of general calculations. We use the notation \([m, n] = \{x \in \mathbb{N} : m \leq x \leq n\}\), and also \([n] = [1, n]\).

Without loss of generality, we refer to the biproduct or direct sum of some objects \(A_1, \ldots, A_n\) as a specific object \(\oplus_{i=1}^n A_i\) with injections \(i_1, \ldots, i_n\) and projections \(\pi_1, \ldots, \pi_n\) such that

\[\pi_i \circ i_j = \begin{cases} 1_{A_i} & i = j \\ 0_{A_i, A_j} & i \neq j \end{cases}\]

where \(0_{A_i, A_j} : A_j \rightarrow A_i\) is the zero morphism.

An arrow \(f : \oplus_{j=1}^n A_j \rightarrow \oplus_{i=1}^m B_i\) can be described uniquely by the set of arrows \(\{f_{ij} : i \in [m], j \in [n]\}\), where for all \(i\) and \(j\)

\[f_{ij} := \pi_i \circ f \circ i_j\]

The separation of fragments of the source and target leads to the matrix representation of \(f\), and can be written

\[f = \begin{bmatrix} f_{11} & \cdots & f_{1n} \\ \vdots & \ddots & \vdots \\ f_{m1} & \cdots & f_{mn} \end{bmatrix}\]

These representations are well behaved with respect to addition, composition, and scalar multiplication, that is, given two arrows \(f\) and \(g\) with appropriate respective sources and targets, and scalar \(s\),

\[(f + g)_{ij} = f_{ij} + g_{ij}\]
\[(g \circ f)_{ij} = \sum_k g_{ik} \circ f_{kj}\]
\[(s \cdot f)_{ij} = s \cdot f_{ij}\]
Being additive and compositional identities for the arrows, the zero and identity morphisms are represented by appropriately-sized zero and identity matrices respectively. When we talk about these matrices in index notation, we write them as 0_I and δ_I (the Kronecker delta) respectively.

When we restrict our interest to arrows between direct sums of the tensor unit I, that is objects of the form nI ⊗ I for some n ∈ N^+, then we are genuinely working with matrices over the semiring C[I, I]. The source and target information of the arrows 0_I and 1_I is discarded for ease of writing. When dealing with arrays whose source and/or target is I, removing the effectively redundant indices is acceptable.

A. Tensor Representation

In naive terms, tensors generalise the concept of matrices. An (n_1 × ⋯ × n_T)-tensor t_i = t_i_1⋯_T is a multidimensional array with T entry indices, the kth of which ranges over [n_k]. We call T the order of t_i. Tensors may be operated on in an analogous manner to matrices, and may also be joined by use of an outer product ⊗: given tensors t_I and v_J, we find (ν ⊗ v)_IJ = u_I ⋅ v_J.

Due to the preservation of products and coproducts by the tensor product, the effectively redundant indices is acceptable.

The notions of the zero and identity matrices are extended so as to offer tensor representations for more complex entities: 0_I⋯_M = ⊗_{k=1}^M 0_{I_k} and δ_{I_1⋯_M} = ⊗_{k=1}^M δ_{I_k}.

Due to the bifunctor Y being equivalent to these arguments also apply to the former, and so the tensor representations can be used to model arrows between instances of all forms of MLL^− functor.

B. Representations of Proof Structures

The tensor representations of formal derivations can be built in a simple manner. The dinatural transformation η for an axiom link is defined by using the family of isomorphisms ν = {ν_R ∈ C[R, R] ≅ C[I, R ⊗ R^−]}_{R ∈ C} on the identity natural transformation on C.

Letting R = nI for some n ∈ N^+, we find that

\[ (ν_I ⊗ (n_J))^{-1} ∘ η ≅ η_I = \begin{cases} 1 & i = j \\ 0 & i ≠ j \end{cases} \]

Because of this, we know the identity rule of MLL^− is modelled by the Kronecker delta δ_I. Continuing further, the unary and binary rules relating to the creation of Y and ⊗ connectives in a sequent are modelled in tensor form by nothing and the outer product respectively. Therefore, each proof net of a sequent containing N positive and N negative literals is modelled by a Kronecker delta δ_{I_1⋯_J_N}, where for each k ∈ [N] the pair of positive and negative literals associated with i_k and j_k are joined by an axiom link in the proof net. In fact, thinking of δ_I as modelling an axiom link, any proof structure can be described using a bijection between the index sets of the positive and negative literals, and so is a tensor of this kind.

With this information, and knowing that MLL^− transformations in feebly fully complete compact closed categories with finite products take the form of linear combinations of interpretations of proof structures, we know that any MLL^− transformation τ over a functor with N positive and N negative arguments is represented by tensors of this form

\[ τ_{I_1⋯_J_N} = Σ_{k=1}^N s_β ∘ δ_{I_1}^{β(1)}⋯_N^{β(n)} \]

Here each β over which we sum is a bijection between the positive and negative index sets, and for each β, s_β belongs to the set of scalars of the category.

C. Multilinear Algebra for Double-glued Categories

Given our category C, for GC the values and covalues of the objects are merely sets of morphisms from I and to ⊥ = 1^− respectively. Therefore, given an object (F([R, R], U, X)) where F is an MLL^− functor of L literals (both positive and negative), we conclude that arrows in both U and X are representable by tensors of order L over C[I, I], as long as each component of R is a direct sum of the tensor unit.

For this reason we study objects of the form F([Z, Z]) for Z ∈ (GC)^N by showing how arithmetic on tensors affects values
and covalues.

\[
(A \otimes B)_{\text{val}} = \{ u_1 \cdot v_1 : u_1 \in A_{\text{val}}, v_1 \in B_{\text{val}} \}
\]

\[
(A \otimes B)_{\text{CoVal}} = \{ v_1 : \exists u_1 \in A_{\text{val}} v_1 u_1 \in B_{\text{CoVal}}, \forall v_2 \in B_{\text{val}} v_1 v_2 \in A_{\text{CoVal}} \}
\]

\[
A_{\bot} = (|A|_{\bot}, A_{\text{CoVal}}, A_{\text{val}})
\]

Note how the tensor representation system allows us to ignore in principle much of the workings of the negation functor when we restrict ourselves to objects generated from direct sums of the tensor unit. As such, any self-dual object in \( \mathbf{GC} \) over \( nI \) for some \( n \in \mathbb{N}^+ \) with its values and covalues represented by tensors can be viewed not just as isomorphic to its dual, but as fundamentally the same.

**VII. Higher-Order Full and Partial Permutations**

It is well known that a permutation on a finite set \( [n] \) can be written in the form of an \( n \times n \)-matrix in which each row and column contains exactly one non-zero entry, namely 1. We extend this notion to give permutations of higher orders: tensors in which there is exactly one non-zero entry in every column. Similarly, we discuss the concept of permutation tensors which are incomplete in the sense some of their columns contain only zero entries.

**Definition 4.** A partial T-permutation over \( [n] \) is an \( n^T \)-tensor \( p_{i_1 \ldots i_l} \) such that for all \( k \in [T] \), and \( x_k \in [n] \) for each \( l \neq k \), there exists an \( x_k \in [n] \) so that \( p_{i_1 \ldots i_l} \cdot \prod \delta_{x_l} = \delta_{x_k} \) or 0\(_{x_k}\). If the former equality is always true we call \( T \) a (full) permutation. We use the notation \( \text{Perm}(T,n) \) (\( \text{PPerm}(T,n) \)) to denote the set of all (partial) \( T \)-permutations over \( [n] \).

Note that a 1-permutation over \( [n] \) is simply a tensor of the form \( \delta_{x} \) for some \( x \in [n] \), and a 2-permutation is a standard permutation.

Not every partial permutation can be completed to a permutation. For example, a tensor \( p_{i_1j_1} \in \text{PPerm}(3,3) \) defined such that \( p_{11} = p_{12} = p_{23} = 1 \) cannot be extended by changing some non-zero entries to have value 1 to give a tensor \( p_{i_1j_1} \in \text{Perm}(3,3) \). This of course means that some work is necessary to show that one can create a higher-order permutation when given a partial one, and we impose suitable restrictions below.

**A. Occurrences of Permutations**

Full and partial permutations can be observed in tensor representations of values and covalues of certain \( \mathbf{GC} \)-objects. We define two kinds of test object that reappear below.

For a non-zero natural number \( n \), let the \( \mathbf{GC} \) object \( A_n \) be given by \((nI, U \cup \{0\}, U \cup \{0\})\), where \( U = \{ \delta_{x} : x \in [n] \} \).

One can show \(A^\otimes T_n = \bigotimes_{i=1}^T A_n = (\otimes_{i=1}^T (nI), V \cup \{0_{1 \ldots i} \}, \text{PPerm}(T,n)) \) where \( V = \{ \delta_{x} \} : \forall i \in [T], x_i \in [n] \} \). The set of covalues for the tensor product above is the collection of higher-order partial permutations of the appropriate order and size.

Similarly, we can show that the covalues of the tensor products of \( B_n = (nI, U, U) \) are full higher-order permutations.

\[
B_{n}^\otimes T = \bigotimes_{i=1}^T B_n = (\bigotimes_{i=1}^T (nI), V, \text{Perm}(T,n))
\]

We can utilise this knowledge when dealing with MDNF functors. Suppose that \( F \) is in MDNF, and therefore if we let \( A_n = (A_1, \ldots , A_n) \),

\[
F(A_n, A_n) = \mathcal{P}(\mathbb{M}^n_{m=1} \bigotimes_{i=1}^T A_{\phi(m,i)}).
\]

As discussed in Section VI-A, the tensor representation does not differentiate between an object and its dual if the object is self-dual, as \( A_n \) is for any number \( n \). This means that the values and covalues of the above functor are the same as if the function \( \phi \) were omitted. We therefore come to the conclusion that, letting \( P_m \) and \( N_m \) be the number of positive and negative literals in block \( m \) respectively, the set of values \( F(A_n, A_n) \) takes the form given below.

\[
\mathcal{P}(\mathbb{M}^n_{m=1} \bigotimes_{i=1}^T A_{\phi(m,i)}).
\]

In other words, an arrow in the homset \( \mathcal{C}[I,F(A_n, A_n)] \) does not lift to \( \mathcal{C}[I,F(A_n, A_n)] \) if its tensor representation, when composed with partial permutations for all bar one of the modelled sequent’s blocks, results in a tensor containing more than one non-zero entry or an entry not equal to either 0 or 1.

Using exactly the same argument, for \( B_n = (B_1, \ldots , B_n) \) we find that \( F(B_n, B_n) \) is \( \mathcal{P}(\mathbb{M}^n_{m=1} \bigotimes_{i=1}^T B_{\phi(m,i)}) \).

An arrow in \( \mathcal{C}[I,F(B_n, B_n)] \) cannot be lifted if its tensor representation, when composed with full permutations for all bar one of the modelled sequent’s blocks, results in a tensor which does not have exactly one non-zero entry or the sole non-zero entry is not equal to 1.

**VIII. Proving Full Completeness for \( \mathbf{GC} \)**

We assume from now on that \( \mathcal{C} \) is a compact closed category with finite biproducts satisfying feeble full completeness, and it is our intention to prove that the double-glued category \( \mathbf{GC} \) satisfies full completeness for \( \text{MLL}^- \). We first prove that this category satisfies MDNF full completeness. Once this has been achieved the more general result follows through standard properties of \( * \)-autonomous categories.

**A. MDNF Full Completeness for \( \mathbf{GC} \)**

In order to prove that \( \mathbf{GC} \) satisfies MDNF full completeness, we follow a set *modus operandi*. In Sections V and VI-B it is illustrated that \( \text{MLL}^- \) transformations in \( \mathbf{GC} \) have a prescribed form. We show that every MDNF transformation
\( \tau = (t_\mathcal{R} \in \mathbb{C}(I, |F([\mathcal{R}, \mathcal{R}])|_{\mathcal{R} \in \mathbb{C}}) \) in \( \mathcal{C} \) representing a linear combination of MDNF proof structures for the sequent modelled by \( F \) which is not a singular proof net cannot be lifted to a transformation in \( \mathcal{G} \mathcal{C} \). This is done by choosing a \( \mathcal{G} \mathcal{C} \)-object, say \( Z = (n1, U, X) \), for some \( n \in \mathbb{N}^+ \) such that \( t_\mathcal{R} \notin F([Z, Z]) \), where \( \mathcal{R} = (n1, \ldots, n1) \) and \( Z = (Z, \ldots, Z) \). We use the objects \( A_n, B_n \in \mathcal{G} \mathcal{C} \) from Section VII-A for certain values of \( n \), and we utilise the tensor representations of the values and covalues of these objects in order to simplify the mathematics.

We work in an unusual order: rather than proving that singular proof structures are the only types of linear combination which are modelled then showing correctness as necessary as well, we do the reverse. We call a linear combination of proof structures trivial if it is one proof structure multiplied with the scalar 1, and non-trivial otherwise. It is first demonstrated that cyclic proof structures cannot be contained in modelled linear combinations, then disconnectness is shown to be forbidden. Only then do we prove that non-trivial linear combinations and scalar multiples are not represented by transformations in \( \mathcal{G} \mathcal{C} \). Before each lemma a corresponding algorithm is given. The algorithms provide full and partial permutations whose existence make possible the proofs that certain liftings of MDNF transformations from \( \mathcal{C} \) into \( \mathcal{G} \mathcal{C} \) do not exist.

**Algorithm 5.** Input: A linear combination of MDNF proof structures of the same sequent containing \( M \) blocks, at least one of which is cyclic.

Output: A number \( n \in \mathbb{N}^+ \); tensors \( a^1, \ldots, a^M \) such that \( a^m_{i_1 \ldots i_m} \in \text{PPerm}(T_m, n) \) for each \( m \).

1. Let \( \lambda \) be the set of axiom links which, with the structure, produces a cyclic maximal \( \mathfrak{T} \)-free subgraph. Note that, at this point, none of the links in \( \lambda \) has been dealt with. We name the set of axiom links used as edges within a chosen minimal length cycle \( \hat{\lambda} \subseteq \lambda \), and let \( i = 0 \).

2. Find a link \( l \in \hat{\lambda} \setminus \hat{\lambda} \) which has not been dealt with yet.
   a. If one should exist, then assign the number \( i+1 \) to both literals incident to \( l \). Increment \( i \), and go to Step 2.
   b. If one does not exist, then move to Step 3.

3. Find a link \( l \in \hat{\lambda} \) which has not been dealt with yet.
   a. If one should exist, then assign both the numbers \( i+1 \) and \( i+2 \) to both literals incident to \( l \). Increase the value of \( i \) by 2, and restart Step 3.
   b. If one does not exist, then move to Step 4.

4. For each tensor product of literals which does not contain a literal incident to a link within \( \hat{\lambda} \), place the values assigned to each literal into a tuple in the same order as their literals appear in the subformula. This tuple ‘belongs’ to that subformula.

5. For each tensor product of literals which does contain a literal incident to at least one link within \( \hat{\lambda} \), create two tuples as follows:
   a. Place the lowest values assigned to each literal into a tuple in the same order as their literals appear in the subformula.
   b. Place the highest values assigned to each literal into a tuple in the same order as their literals appear in the subformula.

6. Set \( n = i; \) and for each \( m \in [M] \), define an element \( a^m_{i_1 \ldots i_m} \) of \( \text{PPerm}(T_m, n) \) as follows:
   \[
   a^m_{i_1 \ldots i_m} = \begin{cases} 
   1 & \text{if } (i_1, \ldots, i_{m_1}) \text{ is a tuple for block } m \\
   0 & \text{otherwise.}
   \end{cases}
   \]

This algorithm creates partial permutations for each block in the input MDNF sequent in such a way that for tuples \((i_1, \ldots, i_{m_1}) \) and \((j_1, \ldots, j_{m_2}) \) with

\[
i_k = j_l \text{ if } k \neq l \text{ if } \mu_1 = \mu_2 \text{ and } a^m_{i_1 \ldots i_{m_1}} = 1 = a^m_{j_1 \ldots j_{m_2}}
\]

the \( \mu^k \) and \( \mu^l \) literals in the \( \mu^k \)th and \( \mu^l \)th blocks respectively must be joined by an axiom link in \( \hat{\lambda} \). Further, if the literals in question are involved in the cycle \( \hat{\lambda} \), then there are two possible values which \( i_k \) and \( j_l \) can take where this situation occurs, and this underpins the proof of the lemma below.

**Lemma 6.** Every MDNF transformation in \( \mathcal{G} \mathcal{C} \) models a linear combination of cyclic proof structures.

**Proof:** Suppose that the functor \( F \) is in MDNF, and therefore for any \( n \in \mathbb{N}^+ \)

\[
F(A_n, A_n) = \mathfrak{T}^M_{n=1} (\mathfrak{T}^m_{t=1} A^t_{\Phi(m,t)}).
\]

The tensor representations of the values of the above object are given in Section VII-A.

Let \( \tau = (t_\mathcal{R} \in \mathbb{C}(I, |F([\mathcal{R}, \mathcal{R}])|_{\mathcal{R} \in \mathbb{C}}) \) be an MDNF transformation in \( \mathcal{C} \), modelling a linear combination of proof structures, one of which is cyclic. For the object \( R = (n1, \ldots, n1) \), the component \( t_\mathcal{R} \) is represented by the tensor

\[
\tau_{i_1(1,1)} \cdots j_{i_{m_1}(M_{m_1})} = \sum_p s_p \cdot \delta^p_{i_1(1,1)} \cdots j_{i_{m_1}(M_{m_1})}.
\]

One of the bijections, \( s \), say, is such that \( s_\zeta \neq 0 \), and there is a cycle between the blocks.

We show that \( t_\mathcal{R} \notin F(A_n, A_n)_{\text{val}} = \mathcal{G} \mathcal{C}(I, F(A_n, A_n)) \) for some choice of \( n \in \mathbb{N}^+ \). We choose \( n \) and define a set of partial permutations \( \{a^m : m \in [M]\} \) one for each block of tensor products of literals, using Algorithm 5; and we nominate any one of the blocks that is part of the chosen cycle, calling it \( k \). We find for the two distinct entry tuples \((x^1, y^1) = (x^1_{(k,1)}, \ldots, x^1_{(k,1)}, \ldots, y^1_{(k,N_k)}) \) and \((x^2, y^2) = (x^2_{(k,1)}, \ldots, x^2_{(k,1)}, y^2_{(k,1)}, \ldots, y^2_{(k,N_k)}) \) in \( \mathfrak{D}^2 \) where \( \mathfrak{D}^2 = 1 = \mathfrak{D}^2_{a^1a^2} \) that, for \( l \in \{1, 2\} \)

\[
\delta^l_{i_1(1,1)} \cdots j_{i_{m_1}(M_{m_1})} \cdot \tau^1_{i_1(1,1)} \cdots j_{i_{m_1}(M_{m_1})} \cdot \prod_{l \notin k} (a^m_{l(i_{m_1}+1)} \cdots j_{i_{m_1}(M_{m_1})}) = s_\zeta \neq 0.
\]

The tensor \( t_\mathcal{R} \) when multiplied with these partial permutations, produces a tensor with at least two non-zero entries, meaning it fails the criterion desired of it to belong to \( F(A_n, A_n)_{\text{val}} \). As such \( t_\mathcal{R} \) cannot lift to an arrow in \( \mathcal{G} \mathcal{C}(I, F(A_n, A_n)), \) and consequently \( \tau \) cannot lift to an \( \text{MLL}^+ \) transformation in \( \mathcal{G} \mathcal{C} \).
The lemma above shows that no proof structure occurring within a linear combination of proof structures being modelled in GC can contain a cycle, and therefore the number of edges that exist in the switchings of proof structures which may be modelled in the category must be strictly less than the number of vertices. Given a set of proof structures over the same sequent, we know that all the switchings of all the structures have the same number of edges: every structure contains the same number of links, and every switching of a structure has the same number of edges. An acyclic graph is also connected if and only if the number of edges within it is exactly one less than the number of vertices in it. Assuming that all proof structures in linear combination being modelled are acyclic, but one of them is disconnected, we know that the number of edges must be less than the number of vertices minus 1. This means that every other proof structure involved in the linear combination has too few edges to be simultaneously connected and acyclic. We therefore conclude that if a linear combination of acyclic MDNF proof structures over the same MLL\(^{\pm}\) sequent contains one structure failing to be connected, then none of the structures is connected. We use this fact in the algorithm and lemmata below.

**Algorithm 7.** Input: A linear combination of acyclic, disconnected MDNF proof structures of the same sequent containing \(M\) blocks.

Output: A number \(n \in \mathbb{N}^+\); tensors \(d^1, \ldots, d^M\) such that \(d^{i_1 \ldots i_m}_{i_1 \ldots i_m} \in \mathbb{PPerm}(T_m, n)\) for each \(m\).

1) Let \(\{\lambda_1, \ldots, \lambda_K\}\) be the set of axiom links in a linear combination of acyclic yet disconnected MDNF proof structures which are multiplied by a non-zero scalar. For each of the blocks containing exactly one literal (leaves) assign distinct values \(1, \ldots, L\). We assign variables \(v_1, \ldots, v_M\) to each of the \(M\) blocks, with \(v_m = T_m\) as the initial setting. We call these variables **valencies**.

2) Take \(\lambda_\ell\) and the \(\mathcal{Y}\)-free subgraph of the parse tree, and consider the component of the graph containing the lowest numbered component which does not contain block 1. Choose the leaf with the highest number in that component and change its valency to \(0\).

3) Choose the first block \(m\) in the component such that \(v_m = 1\).

   a) If one does exist and it is a leaf, decrement the valency of it and its neighbour. If not already assigned a number, give the number of the leaf to it, which is connected. Restart Step 3.

   b) If one does exist but it is not a leaf, then mark the sole literal not yet allocated a number as an ‘exit’ literal and go to Step 4.

   c) If there are no more blocks of valency 1 and \(k < K\), create a new \(T_m\) tuple \(x\) for each block \(m\) in the component, and let \(x_i = u_i\) for every \(i \in [T_m]\), where \(u_i\) is the label given to the \(i^{th}\) literal of the block. If the \(i^{th}\) literal is the exit of the block, then mark \(x_i\) as an exit entry. Delete duplicate tuples associated with each block, and remove all the labels and marks from all the literals except the leaves. Increment \(k\) and return to Step 2.

   d) Otherwise, terminate the algorithm after declaring that \(n = 2^k - 1\) and stating that for each \(m \in [2, M]\),

\[
d^{i_1 \ldots i_m}_{i_1 \ldots i_m} = \begin{cases} 
1 & \text{if } (i_1, \ldots, i_m) \text{ is a tuple for } m \\
0 & \text{otherwise}.
\end{cases}
\]

4) Check to see whether the numbers given to each of the non-exit literals correspond exactly to those of a tuple \(t = (i_1, \ldots, i_m)\) already associated with the block (meaning that if \(u_i\) is the number given to the \(i^{th}\) literal in the block, that \(u_i = t_i\) for every \(i\) for which \(u_i\) is defined).

   a) If so, then assign the final unused number in \(t\) to the exit literal.

   b) If not, assign \(i\) to the exit literal and increment \(i\).

Assign the number to the literal with which the exit literal shares an axiom link, unless that literal is a leaf and already has been assigned a number. Decrement the valencies of both block \(m\) and the block with which the exit literal of block \(m\) is linked by an axiom link in \(\lambda_q\). Go to Step 3.

In order to prove that disconnected proof structures cannot be modelled, either alone or as part of a linear combination, tensor products over an object \(B_n\) for some \(n \in \mathbb{N}^+\) are required. Consequently, it is not possible to utilise partial permutations as the above algorithm has created. However, the following lemma establishes that it is possible to create suitable full permutations from partial ones created in this way.

**Lemma 8.** Every partial higher-order permutation \(d^{m}_{i_1 \ldots i_m}\) where \(m \in [2, M]\) which is an output tensor of Algorithm 7 can be completed to form a full higher-order permutation \(b^{m}_{i_1 \ldots i_m}\).

**Proof:** It is well known that for any \(T\) and \(n\) it is always possible to define the tensor \(e_{i_1 \ldots i_T} \in \mathbb{PPerm}(T, n)\) where

\[
e_{i_1 \ldots i_T} = \begin{cases} 
1 & \text{if } \sum_{i=1}^{T} i_j \equiv 0 \mod n \\
0 & \text{otherwise}.
\end{cases}
\]

It is also well established that, given any permutation in \(\mathbb{PPerm}(T, n)\), it is possible to produce another permutation in the same set by using a (2)-permutation over \([n]\) on the numbering used on a single index.

Suppose that \(d_{i_1 \ldots i_T} \in \mathbb{PPerm}(T, n)\) is such an output tensor. Then we can refer back to the tuples \(t^1, \ldots, t^k\) from which it was created in the algorithm, and as such we know that for every tuple there is a position with an ‘exit marker’. The tuple development procedure ensures that if the \(p^{th}\) position of the \(k^{th}\) tuple is marked, then \(t^p_j \not\equiv t^p_{k'} \) for all \(k' \neq k\). With this information, we define a set of partial functions in \(\mathbb{N}^+\) \(\{a_l : l \in [T]\}\) in the following manner:

1) Let \(i = 1\), \(k = 1\) and \(l = 1\).

2) If \(a_l(t^p_j)\) has not yet been assigned a value and the tuple position in question was not marked an exit entry in Algorithm 7, then we say that \(a_l(t^p_j) = 2^l\) and then increment \(i\). Go to Step 3.

3) If \(l < T\), increment \(l\) and go to Step 2; otherwise increment \(k\) and set \(l\) back to 1 and go to Step 4.
4) If \( k \leq K \), then go to Step 2; otherwise, go to Step 5.
5) Reset \( k \) and \( l \).
6) If \( \alpha_k(t_k) \) has not yet been assigned a value, then set \( \alpha_k(t_k) = n - \sum_{j \neq k} \alpha_k(t_j) \) and go to Step 7. Otherwise increment \( l \) and repeat Step 6.
7) If \( k < K \), increment \( k \) and let \( l = 1 \), and return to Step 6.

Otherwise, terminate the algorithm.

This algorithm is well defined, and each partial function \( \alpha_k \) is injective with all defined source and target values in \([n]\). As such, it is possible to restrict each of the partial functions to act upon \([n]\), and then to extend them to full 2-permutations over \([n]\). We note that \( \sum_{k=1}^{K} \alpha_k(t_k) = n \equiv 0 \mod n \) for each \( k \in [K] \), meaning that the \( T \)-permutation completes \( d_1 \rightarrow T \) satisfactorily.

**Lemma 9.** Every MDNF transformation in \( GC \) modelling a linear combination of acyclic proof structures is modelling a linear combination of cut-free proof nets.

**Proof:** Let \( F \) be an MDNF formula as in Lemma 6, and then consider \( F(B_n,b_n) \) where \( n \) is the output integer given by Algorithm 7. From Section VII-A we know the criteria the tensor representations of the values of this object must satisfy.

Now let \( \tau = (\tau_k \in C[I,F,R]) \) be an MDNF transformation in \( C \) modelling a linear combination of acyclic proof structures, but assume that one (and therefore every one) of these structures is disconnected. Again, for any integer \( n \), we have the tensor representation

\[
\tau_{(1,1) \cdots n}(M_{[g]}) = \sum_{B} \delta_{B} \cdot \tau_{(1,1) \cdots n}(M_{[g]}) .
\]

Bearing in mind that all proof structures modelled over \( F \) must be disconnected, we use Algorithm 7 and the knowledge garnered from Lemma 8 to obtain a set of (full) permutations \( \{B^i : m \in [2,M]\} \) and a suitable number of \( n \) for our definition of \( B_n \).

Take any set of axiom links described by a bijection, \( \zeta \), such that \( \zeta \neq 0 \), and consider the tensor \( \delta_{\zeta(1,1) \cdots n}(M_{[g]}) \). It is possible to factorise this tensor into a product of Kronecker deltas, one for each component in the switching of \( \zeta \). Multiplying the first of the smaller component tensors not containing the indices concerning the first block, \( \delta_{\zeta(1,1) \cdots n}(M_{[g]}) \), say, with each of the permutations \( B^1(1,1), \ldots, B^M(1,1) \) corresponding to the blocks within the component in question, we find that

\[
\delta_{\zeta(1,1) \cdots n}(M_{[g]}) \cdot \prod_{m=1}^{M} (B^m(1,1) \cdots n(1,1)), = 0 .
\]

Multiplying a zero tensor with any other tensor results in another zero tensor, and we know that

\[
\delta_{\zeta(1,1) \cdots n}(M_{[g]}) \cdot \prod_{m=1}^{M} (B^m(1,1) \cdots n(1,1)), = 0 .
\]

Since \( \tau_{(1,1) \cdots n}(M_{[g]}) \) is merely a linear combination of tensors like \( \delta_{\zeta(1,1) \cdots n}(M_{[g]}) \), and \( \zeta \) was arbitrary, we know that

\[
\tau_{(1,1) \cdots n}(M_{[g]}) \cdot \prod_{m=1}^{M} (B^m(1,1) \cdots n(1,1)), = 0 .
\]

This means that \( \tau_{(1,1) \cdots n}(M_{[g]}) \notin F(B_n,b_n) \), and so \( \tau \) cannot be lifted to become an arrow in \( GC[I,F,B_n,B_n] \). The MLL$^-$ transformation \( \tau \) in \( C \) is therefore unable to be lifted into \( GC \) either.

**Algorithm 10.** Input: A non-trivial linear combination of MDNF proof nets for a sequent containing \( M \) blocks. Output: A number \( n \in \mathbb{N}^+ \); tensors \( c_1, \ldots, c_M \) such that \( c_{i_{1} \cdots i_{m}} \in PPerm(T_{m,n}) \) for each \( m \).

1) Let \( \lambda_1 \) and \( \lambda_2 \) be two distinct sets of axiom links in the given linear combination of MDNF proof structures. We set the valencies \( v_1, \ldots, v_M \) such that \( v_m = T_m \) for all \( m \in [M] \), except for the last block, block \( l \) say, for which \( v_l = 0 \). We let \( i \) and \( k = 1 \).

2) Choose the first block \( m \) with \( v_m = 1 \).

   a) If one does exist, then mark the sole literal not yet allocated a number as an ‘exit’ and go to Step 3.
   b) If there are no more blocks of valency 1, create a new \( T_m \) tuple for each block \( m \) in the component, and let \( x_i = u_i \) for every \( i \in [T_m] \), where \( u_i \) is the label given to the \( i^\text{th} \) literal of the block. Remove all the labels. If \( k = 1 \), increment \( k \) and restart Step 2; otherwise go to Step 4.

3) Check to see whether the numbers given to each of the non-exit literals in the block correspond exactly to those of a tuple \( t = (i_1, \ldots, i_{T_m}) \) already associated with the block.

   a) If so, then assign the final unused number in \( t \) to the exit literal.
   b) If not, increment \( i \) and then assign the new value of \( i \) to the exit literal.

Assign the number to the literal with which the exit literal shares an axiom link in \( \lambda_2 \). Decrement the valencies of both block \( m \) and the block with which the exit literal of block \( m \) is linked by an axiom link in \( \lambda_2 \). Restart Step 2.

4) Declare that \( n = i \), and we state that for each \( m \in [M] \),

\[
c_{i_1 \cdots i_{T_m}} = \begin{cases} 1 & \text{if } (i_1, \ldots, i_{T_m}) \text{ is a tuple for } m \\ 0 & \text{otherwise} \end{cases} .
\]

The algorithm above creates simple partial permutations in a similar method to Algorithm 5, and ensures that \( c_{1, \ldots, 1} = 1 \) if \( k = j \) for \( k \) and \( l \) (\( \mu_1 \neq \mu_2 \)) only if the \( k^\text{th} \) literal of block \( \mu_1 \) and the \( l^\text{th} \) literal of block \( \mu_2 \) share an axiom link in either \( \lambda_1 \) or \( \lambda_2 \). Furthermore, if there are two non-zero entries in a partial permutation, then each holds information about exactly one of the two sets of axiom links; and there are definitely two non-zero entries in the tensor corresponding to the last leaf.
Lemma 11. Every MDNF transformation in GC modeling a linear combination of proof nets is modeling a unique proof net.

Proof: Let $F$ be an MDNF formula, and consider $F(A_n, A_0)$ for arbitrary $n$, where $A_i$ is defined as in Section VII-A. Again, we use the criteria also given in Section VII-A that the tensor representation of an arrow $f \in C[I, F(A_n, A_0)]$ must satisfy in order to lift to GC. Suppose that $\tau$ is an MDNF transformation in $C$ which models a linear combination of two or more proof nets. Then, for $R = (n_1, \ldots, n_I)$, $\tau_R$ once again takes the tensorial form

$$\tilde{\tau}_R^{(1,1)} = \sum_{\nu} \tilde{\tau}^{(1,1)} \prod_{m} (c^m)^{i_m} \cdot (j_{m,nu})$$

and we know that there are two bijections, $\zeta_1$ and $\zeta_2$ say, corresponding to axiom link sets $\zeta_1$ and $\zeta_2$ respectively, where $s_{\zeta_1} \neq 0 \neq s_{\zeta_2}$. We use these two sets of axiom links in Algorithm 10 to produce partial permutations $\{e_m^m : m \in [M]\}$ for all of the blocks, and to fix the required size of the number $n \in \mathbb{N}^+$ to prove the lemma.

Let $x_1$ and $x_2$ be the first and second tuple values associated with the last leaf (block $l$) from the algorithm. Then we find that for $k \in \{1, 2\}$

$$\delta^{s_{\zeta_1} \cdot \tilde{\tau}^{(1,1)} \prod_{m} (c^m)^{i_m} \cdot (j_{m,nu})} = s_{\zeta_k} \neq 0$$

where the index $r_{(1,1)}$ equals $i_{(1,1)}$ or $j_{(1,1)}$ depending on whether the literal of $l$ is positive or negative.

We have shown that there are at least two non-zero entries in $\tau^{(1,1)} \prod_{m} (c^m)^{i_m} \cdot (j_{m,nu})$, and so $\tilde{\tau}_R \notin F(A_n, A_0)_{Val}$. The arrow $\tau_R \in C[I, F(A_n, A_0)]$ does not therefore lift into $C[I, F(A_n, A_0)]$, and so $\tau$ is not an MLL$^-$ transformation.

From the above, we know that every MLL$^-$ transformation in GC models a scalar multiple of a cut-free proof net. To prove that proper scalar multiples are unacceptable, we assume that $\tau$ models such a scalar multiple in $C$, and we consider $F(Z, Z)$, where $Z = (I, \{1\}, \{1\})$ and $Z = (Z_1, Z_2)$. The arrow $\tau = \zeta_1$ for some scalar $\zeta_1 \neq 1$ (iff the bijection on positive and negative literals distinguishes the set of axiom links for the proof net in question). This is since, the $(1 \times 1)$-Kronecker delta is equivalent to the unitary tensor $I$.

Suppose that, for $i \in \{1, 2\}$, $Z_i = (Z_i, \{1\}, \{1\})$, where $|Z_i| \equiv 1$. Then

$$Z_1^{\zeta_1} = (Z_1^{\zeta_1}, \{1\}, \{1\})$$
$$Z_1 \boxtimes Z_2 = (Z_1^{\zeta_1} \boxtimes Z_2, \{1\}, \{1\})$$
$$Z_1 \otimes Z_2 = (Z_1^{\zeta_1} \otimes Z_2, \{1\}, \{1\})$$

By induction we get from this that $1 \neq s_{\zeta_1} \notin F(Z, Z)_{Val}$ for any formula $F$. Consequently the C-arrow $\tau_1$ does not lift to be in $C[I, F(Z, Z)]$, and so neither does the transformation $\tau$.

Corollary 12. If $C$ is a compact closed category with biproducts satisfying feeble full completeness, then $C$ satisfies MDNF full completeness.

B. Extending to MLL$^-$ Full Completeness

Extending this result to MLL$^-$ full completeness is possible due to the existence of ‘weakening’ natural transformations in all *-autonomous categories. Arguments similar to the one below are seen in other full completeness proofs.

In every *-autonomous category $D$ there are natural transformations

$$w_{LL} = (w_{LL}, w_{LR}, w_{RL}, w_{RR}) : (A \otimes (B \otimes C) \rightarrow (A \otimes B) \otimes C)_{A,B,C \in C}$$
$$w_{LR} = (w_{LR}, w_{RL}, w_{RR}) : (A \otimes (B \otimes C) \rightarrow (A \otimes B) \otimes C)_{A,B,C \in C}$$
$$w_{RL} = (w_{RL}, w_{LL}, w_{RR}) : (A \otimes (B \otimes C) \rightarrow (A \otimes B) \otimes C)_{A,B,C \in C}$$
$$w_{RR} = (w_{RR}, w_{LL}, w_{RL}) : (A \otimes (B \otimes C) \rightarrow (A \otimes B) \otimes C)_{A,B,C \in C}$$

In compact closed $C$ these are canonically isomorphic to appropriate compositions of associativity and symmetry isomorphisms. The tensor representations of these natural isomorphisms are both described solely by Kronecker deltas. As discussed in Section III, if $\tau$ is an MLL$^-$ transformation in $C$, we may compose it with a sequence of natural transformations built from the weakening transformations and the associativity and symmetry isomorphisms of $C$ to produce another such; and $\tau$ lifts to an MLL$^-$ transformation in GC only if the result does.

Additionally, we can always find a composition of natural transformations which preserves the cyclicity of at least one modelled cyclic proof structure in an MLL$^-$ transformation. Similarly, simultaneous acyclicity and disconnectedness of modelled proof structures can be preserved; and non-trivial linear combinations and scalar multiples always beget non-trivial linear combinations and scalar multiples (proofs omitted). With these facts in mind, we prove the stronger full completeness theorem.

Theorem 13. If $C$ is a compact closed category with finite biproducts satisfying feeble full completeness then $C$ satisfies MLL$^-$ full completeness.

Proof: Let $\tau$ be an MLL$^-$ transformation in $C$ modeling a linear combination of proof structures, at least one of which is cyclical. Using a suitable algorithm to generate an appropriate natural transformation $\tilde{\nu}$, we obtain a family of arrows $\tilde{\nu} = \nu \circ \tau$ modeling a linear combination of MDNF proof structures, with at least one of these containing a switching cycle. By Proposition 12, $\tilde{\nu}$ cannot lift to an MDNF transformation in GC; and therefore it immediately follows by the preceding discussion that $\tau$ does not lift either.

Similar arguments apply when $\tau$ models a linear combination of cyclic yet disconnected proof structures, and a linear combination of cut-free proof nets. As such, the family $\tau$ may only lift to an MLL$^-$ transformation in GC if it models one individual proof net.

Corollary 14. For a tensor-generated compact closed category with biproducts $C$ the category $C$ satisfies MLL$^-$ full completeness.

IX. CONCLUSIONS

We have shown that the double glued construction is effective at producing fully complete categorical MLL$^-$ mod-
models from degenerate categorical models. This is a substantial generalisation of [14] which inspired our original conjecture, and the result is generated through combinatorial methods. The approach taken, which focuses on the positioning of non-zero entries in certain tensors rather than the actual scalar values entered, refrains from the use of addition of any two non-zero scalars—this is the key to the theorem’s generality. Furthermore, the methods used to obtain the proof are considerably more ‘concrete’ than those from the material generalised, and lend themselves to finding a closer connection between the original syntactic framework of MLL\(^{-}\) and its semantics. An example of how such connections could manifest themselves can be seen by comparing our primary method for disproving the existence of certain MLL\(^{-}\) transformations with the Counterproof Criterion [4]: Each higher-order permutation models a form of paraproof for the negation of each block. Since the morphisms that describe proofs are represented by certain isotropic tensors, there might also be a purely geometric interpretation of the concepts in this paper, which could lead to an interesting alternative perspective on the logic.

The arguments can also be extended beyond the main theorem given above with relative ease. A proof of MLL\(^{-}\)+Mix full completeness for certain compact closed categories with finite biproducts under an orthogonalised double glueing construction has been obtained via the tensor calculus, and it is thought that the tight correspondence of the category of coherence spaces (\textbf{Coh}) and MLL\(^{-}\) may be explained by similar means.

It is impossible to give full proofs within the page limit, but we have tried to outline the general strategy and given as much technical detail as possible. Certainly the algorithms and some of the lemmata would benefit from having examples but we found it impossible to fit these in. We expect to provide full details in a future journal publication.

One reason why double glueing constructions such as the one used above are of interest is due to their preservation not just of \(\ast\)-autonomy but of products and coproducts, making them perfect for refinement of models of unit-free multiplicative \textit{additive} linear logic (MLL\(^{-}\)). The given glueing construction does not produce a similar full completeness result for MLL\(^{-}\) for compact closed categories \(\mathbf{C}\) with biproducts (at least there is no known case where it does). However, we believe that there may exist another double glueing construction that leads to fully complete models for MLL\(^{-}\). Investigating how an adaptation of the tensor calculus may be implemented to prove such a theorem is ongoing work by the authors.

\textbf{Acknowledgment}

Hugh Steele is supported by an EPSRC studentship.

\textbf{References}