


The Sum-Product Algorithm for Quantitative Multiplicative Linear Logic

Thomas Ehrhard  

Université de Paris Cité, CNRS, IRIF, F-75013, Paris, France <https://www.irif.fr/~ehrhards/>

Claudia Faggian 

Université de Paris Cité, CNRS, IRIF, F-75013, Paris, France <https://www.irif.fr/~faggian/>

Michele Pagani  

Université de Paris Cité, IRIF, F-75013, Paris, France <https://www.irif.fr/~michele/>

Abstract

We consider an extension of multiplicative linear logic which encompasses bayesian networks and expresses samples sharing and marginalisation with the polarised rules of contraction and weakening. We introduce the necessary formalism to import exact inference algorithms from bayesian networks, giving the sum-product algorithm as an example of calculating the weighted relational semantics of a multiplicative proof-net improving runtime performance by storing intermediate results.

2012 ACM Subject Classification Theory of computation \rightarrow Linear logic; Theory of computation \rightarrow Denotational semantics; Mathematics of computing \rightarrow Variable elimination

Keywords and phrases Linear Logic; Proof-Nets; Denotational Semantics; Probabilistic Programming

Digital Object Identifier 10.4230/LIPIcs.FSCD.2023.4

Acknowledgements This work was partially supported by ANR PRC project PPS ANR-19-CE48-0014

1 Introduction

Linear logic [18] provides a linear algebra flavour to logic, associating linear algebra operations with logical connectives, e.g. tensor \otimes is seen as a form of conjunction, direct sum \oplus as a disjunction and duality as an involutive negation $(\cdot)^\perp$. This perspective has given many insights. In denotational semantics, we have *quantitative semantics*, e.g. [25, 21, 10, 11, 6, 24]: a family of models denoting λ -terms and functional programs with some notion of analytic maps or power series that can be locally approximated by multilinear functions, these latter denoting linear logic proofs. In proof-theory, we have *proof-nets*: a representation of proofs and programs expressing the interdependences of these algebraic operations in a graph-theoretical fashion.

Quantitative semantics turns out to be particularly suitable for probabilistic programming, giving fully abstract semantics [14, 15, 17], denoting probabilistic programs with very regular functions (absolutely monotone) even on “continuous” datatypes (e.g. real numbers) [16, 5, 13], giving a compositional analysis of various operational behaviours, such as runtime or liveness [24], providing suitable notions of program metrics [12]. Due to this expressivity, calculating the quantitative denotations for a Turing complete programming language is obviously non-computable, but we can fix on relevant fragments supporting an effective procedure. Effectiveness is a relevant feature for a denotational model, as it can provide automatic tools for verifying program correctness, as well as the other mentioned operational properties.

Let us focus our attention to one of the simplest fragments of linear logic: the multiplicative fragment (MLL), which has the \otimes conjunction, its unit 1 and their respective duals, the par \wp (a disjunction different from \oplus) and $\perp = 1^\perp$. From a programming perspective, this fragment contains (although it is not restricted to) an exponential-free fragment of the linear λ -calculus



© Thomas Ehrhard, Claudia Faggian and Michele Pagani;
licensed under Creative Commons License CC-BY 4.0

8th International Conference on Formal Structures for Computation and Deduction (FSCD 2023).

Editors: Marco Gaboardi and Femke van Raamsdonk; Article No. 4; pp. 4:1–4:23

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

with tuples, e.g. [1]: the linear functional type $F \multimap G$ is in fact represented by $F^\perp \wp G$. Although very simple, this fragment is already surprisingly expressive on probabilistic data. First, positive types (i.e. combinations of 1 , \oplus and \otimes) express linear combinations of the values of a finite data-type. For example, the quantitative denotation of $1 \oplus 1$ contains linear combinations of booleans and can be used to model boolean random variables¹. Moreover, it is known since the inception of polarised linear logic that positive formulas are endowed with a polarised version of the structural rules of weakening and contraction ([19] and Remark 3), so one can represent λ -terms having multiple occurrences of a same boolean variable without breaking the linearity features of MLL. In probabilistic programming, these occurrences duplicate the *samples* from a random variable, but not the random variable itself. Finally, we can allow semantical boxes expressing matrices indexed by finite data-types, which can express conditional probabilities. We call this system *quantitative MLL* (Section 2).

As for the semantics, let us focus on the $\mathbb{R}_{\geq 0}$ -weighted relation semantics (see Section 3 and [24]), which is one of the most basic examples of quantitative semantics, allowing to model probabilistic programs over countable data-types. The denotation of a proof-net is then a vector of dimension equal to the number of the possible samples of a probabilistic distribution computed by the proof-net. This vector is computable for quantitative MLL and the standard semantical definitions yield a recursive procedure (Figure 1c) to compute it. In practice, this procedure is unfeasible, as it is exponential in time and in space with respect to the size of the proof-net. The goal of this paper is to inaugurate a new approach for improving it by taking inspiration from bayesian networks, which have partially a similar graph-theoretical structure as proof-nets.

For example, the $\mathbb{R}_{\geq 0}$ -weighted denotation of a proof-net describing a probabilistic distribution over a tuple of n booleans is a vector of dimension 2^n (the number of the possible outcomes of a random variable over n booleans), independently whether the values of some of these booleans depend each other or not (Example 9). The proof-net carries very clearly these interdependences via paths over boolean edges: may we reduce the dimension of its denotation by following such a structure? On a different note, the composition of two proof-nets on a tuple of n booleans yields a sum of 2^n terms (Example 11). However, this composition can be ordered by following the switching paths over the corresponding cuts. May we refactor the sum according to this order and gain in efficiency by memorising some intermediate factors?

Similar questions are typical of the research on Bayesian networks ([27], see as reference [8]), these latter being directed graphs expressing the conditional dependences between different random variables. The benefit of this approach is to provide a battery of algorithms computing, e.g., marginal distributions in a quite efficient way by taking advantage of the graph-theoretical structure of a network. Our general goal is to inaugurate a new approach to quantitative semantics which pays attention to the cost of computing the semantics, and we do so by exploiting techniques from Bayesian inference. One paradigmatic example is the *sum-product variable elimination algorithm* [29]: we propose here a formalism for computing the semantics of a quantitative MLL proof-net by adapting this algorithm (here Algorithm 1).

Related works. Bayesian networks form, *mutatis mutandis*, a strict subset of quantitative MLL proof-nets (Remark 1), morally the set of those proof-nets which do not contain

¹ It is known that the space of random variables ranging over a *finite* set of outcomes of cardinality n can be described by the finite additive disjunction $\bigoplus_{i \leq n} 1$ of the tensor unit, see e.g. [17]. This formula is not in MLL, as \oplus is not a multiplicative connective, but it appears in our setting as these spaces of finite random variables are associated with the positive atomic formulas of MLL (see Example 7).

87 formulas alternating polarities, e.g. alternation of \otimes and \wp connectives. This correspondence
 88 has been already acknowledged, with a slight different terminology, by the recent literature
 89 about the semantical foundations of Bayesian programming. We mention in particular
 90 [3, 22] which represent Bayesian networks as string diagrams and analyse the notion of
 91 disintegration. The paper [26] proposes a game semantics based on event structures for a
 92 variant of the linear λ -calculus underlined by quantitative MLL. The paper [28] studies an
 93 equational theory and provides a denotational semantics based on matrices for this calculus
 94 when restricted to ground data-types. However, to our knowledge, our paper is the first time
 95 that the efficiency of computing the semantics is taken into consideration. Moreover, we show
 96 that the techniques of Bayesian networks can be adapted to the more general framework of
 97 quantitative MLL without so much effort.

98 **Paper outline.** Section 2 introduces quantitative MLL proof-nets and Section 3 its
 99 associated $\mathbb{R}_{\geq 0}$ -weighted relational semantics. Section 4 revisits the standard notion of factor
 100 in Bayesian inference so to apply it to atomic proof-nets in Section 5 and to general proof-nets
 101 in Section 6. Section 7 concludes by mentioning some future developments.

102 2 Quantitative Multiplicative Linear Logic

103 Metavariables X, Y, Z will vary over a countable set of propositional variables. The grammar
 104 of the formulas of MLL is given by (together with its metavariables):

$$105 \quad F, G, H ::= X^+ \mid X^- \mid 1 \mid \perp \mid F \otimes G \mid F \wp G. \quad (1)$$

107 We call X^+ (resp. X^-) a *positive atomic formula* (resp. *negative atomic formula*) over the
 108 variable X , the superscript symbol $+$ (resp. $-$) being its *polarity*. We will write X° for a
 109 generic atomic formula over X , if we do not want to precise its polarity. The linear logic
 110 negation is introduced as syntactical sugar: $(X^+)^\perp ::= X^-$, $1^\perp ::= \perp$, $(F \otimes G)^\perp ::= F^\perp \wp G^\perp$,
 111 and for the dual cases (X^-, \perp, \wp) , $(F^\perp)^\perp ::= F$.

112 A *sequent* is a finite sequence F_1, \dots, F_n of MLL formulas. Capital Greek letters Γ, Δ, \dots
 113 will vary over sequents. Given a sequent $\Gamma = F_1, \dots, F_n$, we write Γ^\perp for the sequent
 114 $F_1^\perp, \dots, F_n^\perp$. Moreover, if $n > 0$, we write $\wp\Gamma$ (resp. $\otimes\Gamma$) for the formula $F_1 \wp (\dots \wp F_n)$
 115 (resp. $F_1 \otimes (\dots \otimes F_n)$). If Γ is empty (i.e. $n = 0$), $\wp\Gamma$ (resp. $\otimes\Gamma$) will mean \perp (resp. 1).

116 As accustomed in linear logic, sequent proofs are represented by special graphs, called
 117 *proof-nets*. Figure 1e gives an example of two proof-nets: \mathcal{N} at the left side of the arrow
 118 $\xrightarrow{*}$, and \mathcal{N}_0 at the right side. A proof-net is a labelled directed acyclic graph² (DAG for
 119 short) such that the edges are labelled by MLL formulas and the nodes by deduction rules of
 120 our extended MLL, i.e. by a symbol among: **ax** (axiom), **cut** (cut), **1** (one), \otimes (tensor), \perp
 121 (bottom), \wp (par), **w** (weakening), **c** (contraction), **b** (semantical box or simply box). The
 122 nodes of the proof-nets in Figure 1e are represented just by their labels, except for the box
 123 which is depicted as a rectangular and labeled by an enumerated occurrence of **b**. The label
 124 of a node determines the number of incoming edges (called *premises* of the node) and the
 125 number of outgoing edges (called *conclusions* of the node), as well as the type of formulas
 126 labelling these edges, according to the rules sketched in Figure 1a. The edges will be oriented
 127 top-bottom, so that axioms, ones, bottoms, weakenings and boxes have no premises, while

² More formally, a directed graph is a quadruplet $(V, E, \mathbf{t}, \mathbf{s})$ of a set V of vertices and a set E of edges, and two maps $\mathbf{t}, \mathbf{s} : E \mapsto V$ associating an edge with a target and a source, respectively. We allow directed graphs with pending edges, i.e. \mathbf{t} and \mathbf{s} may be partial partial. The edges not in the domain of \mathbf{t} or \mathbf{s} are called *pending*. A directed graph is acyclic (a DAG for short), if there is no directed cycle.

cuts have no conclusions. Figure 1e does not explicit all formulas labelling the edges of \mathcal{N} and \mathcal{N}_0 , in fact these formulas can be recovered by the axioms and boxes labelling and the rules sketched in Figure 1a. Proof-nets have edges without targets which are called *the conclusions of the proof-net*. Both \mathcal{N} and \mathcal{N}_0 have one single conclusion, labelled by $X_4^+ \otimes X_5^+$.

Not all DAGs of MLL nodes are proof-nets: the set of *proof-nets* is the subset of the set of all DAGs which can be generated inductively by the rules sketched in Figure 1b. We call *atomic* a proof-net whose edges are only labelled with atomic formulas. Notice that atomic proof-nets can contain only axioms, cuts, weakening, contractions and semantical boxes.

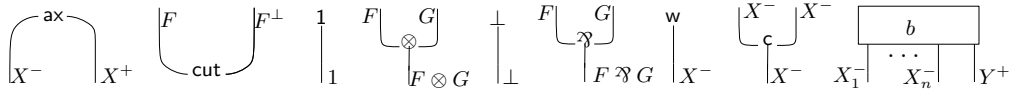
► **Example 1.** The (atomic) proof-net \mathcal{N}_0 in Figure 1e is mutatis mutandis an example of a Bayesian network as expressed by quantitative MLL. The propositional variables X_1, \dots, X_5 are place-holders for (sets of the possible outcomes of) random variables and the semantical boxes are place-holders for their associated “conditional probabilistic tables” (we borrow here the terminology of [8]). For example, the box b_4 is a place-holder for a probabilistic distribution over the variable X_4 conditioned by the outcomes of the variables X_2 and X_3 . The polarities discriminate between input and output occurrences in a conditional probabilistic table. These place-holders will be instantiated with concrete conditional distributions by the semantics, as detailed in Section 3.

The acquainted reader in Bayesian graphs should be convinced that these latter are depicted plainly in this syntax just by adding cuts transforming outputs into inputs. Notice that by inverting the orientation of the edges labelled by negative atoms, we get exactly the same directed paths between the nodes of the corresponding Bayesian network. Of course, MLL allows for more nets than Bayesian graphs, for example the proof-net \mathcal{N} at left of the $\xrightarrow{*}$ arrow is not bayesian, namely it has par nodes. But yet, Remark 54 will allude to a correspondence between \mathcal{N} and a run of the sum-product algorithm over \mathcal{N}_0 . Our goal is to show how Bayesian graph algorithms can be imported in this more general setting.

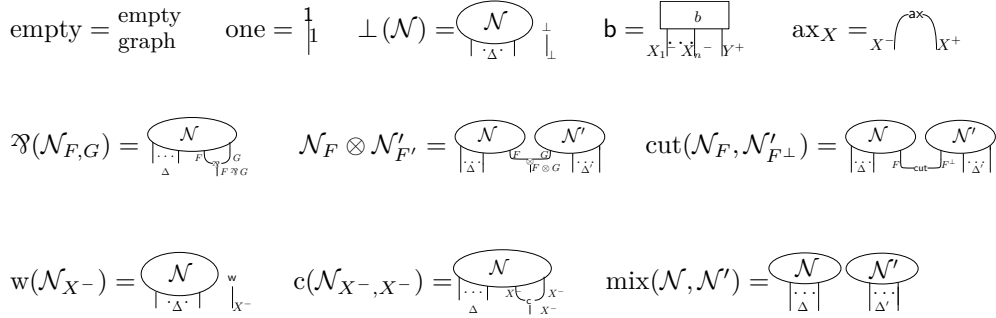
► **Remark 2.** Some papers, e.g. [3, 22], represent Bayesian graphs as string diagrams, which is a graphical syntax omitting the axiom and cut nodes. Although one can present MLL in a similar way by using Lafont’s interaction nets [23], we prefer to keep axioms and cuts explicit as they condense the main threats to an efficient computation of the semantics which is a core topic of this paper.

► **Remark 3.** We allow for structural rules (weakening and contraction) on negative atomic formulas. In fact, as it will be clear in Section 3, negative atoms will be interpreted by finite products of bottoms $\&_{x \in S} \perp$ (although we do not detail here the additive connectives $\&$ and \oplus and the exponential modalities $?$ and $!$). It is well-known since the inception of polarized linear logic [19] that \perp is isomorphic to the exponential formula $?0$, so that the structural rules of $?$ can be lifted to $\&_{x \in S} \perp$, extending the expressivity of MLL. One might even allow the structural rules to all formulas of negative polarity, but we preferred to restrict to atomic formulas to ease the presentation, namely cut reduction.

► **Remark 4.** A less standard extension is given by the semantical boxes b , which are place holders for conditional distributions or, more generally, matrices. For technical convenience, we restrict their conclusions (as well as those of MLL axioms) to be atomic formulas with exactly one occurrence of a positive formula. The structural rules of contractions and weakenings take then a precise operational meaning: a cut between the positive conclusion of a box and a contraction duplicates the *samples* of the probabilistic distribution associated with the box, while weakenings maginalise out this distribution. These operations are categorically axiomatised by the so-called *CD-structure*, for “copier” and “discarder”, e.g. [22, 28]. We



(a) Labelling of MLL nodes with their incident edges. Edges are oriented top-down.



(b) Sequent rules generating the set of proof-nets. The notation \mathcal{N}_Γ in the subscript of a rule stands for the pair of a proof-net \mathcal{N} and a sequence Γ of conclusions of \mathcal{N} , which will be “active” in the rule.

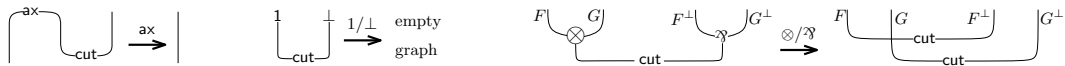
$$\llbracket \text{empty} \rrbracket_\star = \llbracket \text{one} \rrbracket_\star = 1 \quad \llbracket \perp(\mathcal{N}) \rrbracket_{(\vec{d}, \star)} = \llbracket \mathcal{N} \rrbracket_{\vec{d}} \quad \llbracket \mathbf{b} \rrbracket_{(\vec{x}, y)} = \iota(\mathbf{b})_{(\vec{x}, y)} \quad \llbracket \text{ax}_X \rrbracket_{x, x'} = \delta_{x, x'}$$

$$\llbracket \wp(\mathcal{N}) \rrbracket_{(\vec{d}, (x, y))} = \llbracket \mathcal{N} \rrbracket_{(\vec{d}, x, y)} \quad \llbracket \mathcal{N} \otimes \mathcal{N}' \rrbracket_{(\vec{d}, \vec{d}', (x, y))} = \llbracket \mathcal{N} \rrbracket_{(\vec{d}, x)} \llbracket \mathcal{N}' \rrbracket_{(\vec{d}', y)}$$

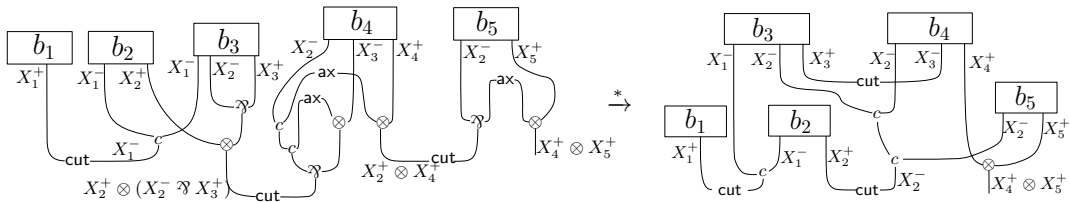
$$\llbracket \text{cut}(\mathcal{N}, \mathcal{N}') \rrbracket_{\vec{d}, \vec{d}'} = \sum_{x \in [F]} \llbracket \mathcal{N} \rrbracket_{\vec{d}, x} \llbracket \mathcal{N}' \rrbracket_{\vec{d}', x}$$

$$\llbracket \mathbf{w}(\mathcal{N}) \rrbracket_{(\vec{d}, x)} = \llbracket \mathcal{N} \rrbracket_{\vec{d}} \quad \llbracket \mathbf{c}(\mathcal{N}) \rrbracket_{(\vec{d}, x)} = \llbracket \mathcal{N} \rrbracket_{(\vec{d}, x, x)} \quad \llbracket \text{mix}(\mathcal{N}, \mathcal{N}') \rrbracket_{(\vec{d}, \vec{d}')} = \llbracket \mathcal{N} \rrbracket_{\vec{d}} \llbracket \mathcal{N}' \rrbracket_{\vec{d}'}$$

(c) Inductive definition of the interpretation $\llbracket \mathcal{N} \rrbracket^\iota$ by induction on a sequence of sequent rules giving \mathcal{N} , we omit to explicit the valuation ι as well as the active sequent in the sequent rule.



(d) MLL cut-reduction rewriting steps.



(e) Example of two proof-nets of conclusion $X_4^+ \otimes X_5^+$ such that $\mathcal{N} \xrightarrow{\star} \mathcal{N}_0$. The labelling of some edges is omitted.

■ **Figure 1** The proof-net syntax and weighted-relational semantics of quantitative MLL.

174 explicit here how the structural polarised linear logic rules perfectly fulfil this rôle, showing
 175 a natural Curry-Howard correspondence with Bayesian programming.

176 Given a proof-net \mathcal{N} and an edge e of \mathcal{N} , we write by $e : F$ whenever e is labelled by the
 177 formula F . By ease of notation, we often write the sequent F_1, \dots, F_n synonymously for an
 178 enumeration $e_1 : F_1, \dots, e_n : F_n$ of labelled edges, if the edges e_1, \dots, e_n are clear from the
 179 context or inessential. We write $\mathcal{N} : \Delta$ whenever the sequent Δ enumerates the (formulas
 180 labelling the) conclusions of \mathcal{N} , also speaking about Δ as simply the conclusions of \mathcal{N} .

181 *Cut-reduction* is defined as a graph-rewriting, replacing a subgraph containing a cut (the
 182 *redex*) with a new subgraph (the *contractum*) having the same pending edges. Figure 1d
 183 sketches the three different kinds of MLL redexes: \mathbf{ax} , $1/\perp$, \otimes/\wp . We will write $\mathcal{N} \rightarrow \mathcal{N}'$
 184 if \mathcal{N} rewrites into \mathcal{N}' by one single rewriting step. The fact that \mathcal{N}' is still a proof-net is
 185 proven by using the so-called correctness criteria (see [18] for details). We denote by $\xrightarrow{*}$ the
 186 reflexive and transitive closure of \rightarrow . A *normal form* is a proof-net which contains no redex
 187 of any kind $\{\mathbf{ax}, 1/\perp, \otimes/\wp\}$. Cut-reduction is confluent and strong normalising [18].

188 ► **Example 5.** Figure 1e gives an example of a proof-net \mathcal{N} that rewrites into the normal
 189 form \mathcal{N}_0 . Notice that cuts between structural nodes (weakening and contraction) and boxes
 190 are not reduced (see Remark 6) so that the normal form \mathcal{N}_0 yet contains some cuts. Notice
 191 also that different sequences of rewriting steps may start from \mathcal{N} but all of them can be
 192 eventually completed into \mathcal{N}_0 , in accordance with the confluence property.

193 ► **Remark 6.** Weakening and contraction do not erase nor duplicate semantical boxes as this
 194 rewriting would break the correspondence with Bayesian networks mentioned in Example 1.
 195 In fact, if we rewrote a cut between a contraction and a box \mathbf{b} into two distinct copies of \mathbf{b} ,
 196 then this would correspond to create two independent and identically distributed random
 197 variables out of a single one and not to duplicate a sample of this latter. The sharing nodes in
 198 bayesian networks share samples of random variables but do not duplicate random variables
 199 (see [7]). We will discuss this point also in Example 8 using the weighted relational semantics.

200 3 Weighted Relational Semantics

201 The quantitative semantics of linear logic refers to a family of denotational models based
 202 on linear algebra constructions (tensors, linear functions, direct sums, dual spaces, etc.).
 203 Many examples are known in the literature, such as finiteness and Koethe spaces [11, 10],
 204 weighted relations [24], probabilistic coherence spaces [6], coherent Banach spaces [20] etc.
 205 The common idea is to associate types with a mathematical structure underlying a notion of
 206 vector space (or a module) and the proofs with linear maps represented by matrices, or simply
 207 vectors in case of proof-nets. We consider here one of the most basic examples of quantitative
 208 semantics, the “relations” weighted by non-negative real numbers, but the results of this
 209 paper can be adapted trivially to any quantitative semantics mentioned above.

210 The model of $\mathbb{R}_{\geq 0}$ -weighted relations is a variant of the relational semantics of linear
 211 logic (see e.g. [2]), where the notion of a subset of a set S , seen as a vector $(b_x)_{x \in S}$
 212 booleans expliciting whether an element $x \in S$ belongs or not to the subset, is generalised to
 213 a vector of non-negative real numbers. This model is known and we thus just sketch here the
 214 interpretation of quantitative MLL proof-nets, referring the reader to [24] for more details.

215 Let us fix some basic notation. Metavariables S, T, U range over finite sets³. We denote
 216 by $\mathbb{R}_{\geq 0}$ the cone of the non-negative real numbers. Metavariables ϕ, ψ, ξ will range over

³ This kind of denotational semantics are defined for countable sets S in general. Infinite sets are necessary

217 vectors in $\mathbb{R}_{\geq 0}^S$, ϕ_x denoting the scalar associated with $x \in S$ by $\phi \in \mathbb{R}_{\geq 0}^S$. The identity
 218 matrix over a set S , also called diagonal matrix or Kronecker delta, is denoted $\delta \in \mathbb{R}_{\geq 0}^{S \times S}$
 219 and defined by $\delta_{a,a'} = 1$ if $a = a'$, otherwise $\delta_{a,a'} = 0$.

220 ► **Example 7.** The simplest example we consider is the singleton set $\{\star\}$, for some irrelevant
 221 element \star . The singleton will be associated with multiplicative units 1 and \perp and it induces
 222 the module of scalars, as $\mathbb{R}_{\geq 0}^{\{\star\}} \simeq \mathbb{R}_{\geq 0}$. The module of couples of non-negative real numbers
 223 is instead induced by any set of cardinality 2, like the set of booleans $\{\mathbf{t}, \mathbf{f}\}$: $\mathbb{R}_{\geq 0}^{\{\mathbf{t}, \mathbf{f}\}} \simeq \mathbb{R}_{\geq 0}^2$.
 224 In all the examples of this paper, we will in fact associate the propositional variables with
 225 the set $\{\mathbf{t}, \mathbf{f}\}$, so that a proof-net with only one atomic conclusion will be interpreted with a
 226 vector $(\lambda_{\mathbf{t}}, \lambda_{\mathbf{f}})$, giving a “score” to the two booleans. Notice that $\mathbb{R}_{\geq 0}^2 = \mathbb{R}_{\geq 0} \oplus \mathbb{R}_{\geq 0}$, with \oplus
 227 denoting the direct sum over modules. This is reflected in linear logic by encoding the type
 228 of booleans with the formula $1 \oplus 1$, where \oplus refers to the additive disjunction. We however
 229 avoid this notation as we do not consider the full additive connectives here.

230 More in general, the interpretation of a MLL formula F is a finite set $\llbracket F \rrbracket^\iota$ defined once
 231 we have fixed a *valuation* ι as a function mapping the propositional variables to finite sets.
 232 The definition of $\llbracket F \rrbracket^\iota$ is by induction on F , as follows:

$$233 \quad \llbracket X^+ \rrbracket^\iota = \llbracket X^- \rrbracket^\iota ::= \iota(X), \quad \llbracket \mathbf{1} \rrbracket^\iota = \llbracket \perp \rrbracket^\iota ::= \{\star\}, \quad \llbracket F \otimes G \rrbracket^\iota = \llbracket F \wp G \rrbracket^\iota ::= \llbracket F \rrbracket^\iota \times \llbracket G \rrbracket^\iota.$$

235 It is easy to check that the usual isomorphisms of linear logic (like associativity and com-
 236 mutativity of the binary connectives) are validated by set isomorphisms. In particular, we
 237 can use tuples (x_1, \dots, x_n) for denoting elements in the interpretation of a n -fold connective,
 238 e.g. $\llbracket F_1 \wp (\dots \wp F_n) \rrbracket^\iota \simeq \{(x_1, \dots, x_n) \mid \forall i \leq n, x_i \in \llbracket F_i \rrbracket^\iota\}$.

239 Weighted relational semantics equates much more than just linear logic isomorphisms,
 240 as for example $\llbracket F \rrbracket^\iota = \llbracket F^\perp \rrbracket^\iota$ for any formula F . More precisely, this semantics has the
 241 structure of a compact closed category. There are more refined examples of quantitative
 242 semantics which are not compact closed, e.g. probabilistic coherence spaces. Let us stress
 243 that our results do not suppose compact closeness.

244 The interpretation $\llbracket \mathcal{N} \rrbracket^\iota$ of a proof-net \mathcal{N} of conclusions Γ is a vector in $\mathbb{R}_{\geq 0}^{\llbracket \wp \Gamma \rrbracket^\iota}$, which
 245 can be equivalently seen as a multidimensional matrix indexed by the tuples in $\llbracket \wp \Gamma \rrbracket^\iota$. The
 246 interpretation can be given inductively as sketched by Figure 1c, once we have associated with
 247 each box \mathbf{b} of conclusions X_1^-, \dots, X_n^-, Y^+ a vector $\iota(\mathbf{b}) \in \mathbb{R}_{\geq 0}^{\llbracket (\wp_i X_i^-) \wp Y^+ \rrbracket^\iota}$. This interpretation
 248 is invariant under the cut-reduction rules of Figure 1d, i.e. $\mathcal{N} \rightarrow \mathcal{N}'$ implies $\llbracket \mathcal{N} \rrbracket^\iota = \llbracket \mathcal{N}' \rrbracket^\iota$.

249 ► **Example 8.** Consider the proof-net \mathcal{N}' of conclusion $X_2^+ \otimes (X_2^- \wp X_3^+)$ contained in the
 250 proof-net \mathcal{N} depicted at left of Figure 1e and characterised by the three boxes $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$
 251 and the tensor and par above the cut over $X_2^+ \otimes (X_2^- \wp X_3^+)$. Notice that there is only one
 252 sequence of the generating rules of Figure 1b producing this proof-net: one first applies a
 253 par rule under the \mathbf{b}_3 conclusions X_2^- and X_3^+ , then a tensor between the resulting proof-net
 254 and \mathbf{b}_2 , then a contraction between the two X_1^- conclusions and finally a cut between the
 255 conclusion of this contraction and \mathbf{b}_1 . Figure 1c applied to this sequence of rules gives:

$$256 \quad \llbracket \mathcal{N}' \rrbracket_{(x', (x'', x''))}^\iota = \sum_{y \in \llbracket X_1^+ \rrbracket^\iota} \iota(\mathbf{b}_1)_y \iota(\mathbf{b}_2)_{(y, x')} \iota(\mathbf{b}_3)_{(y, (x'', x''))}.$$

to model linear logic exponential modality as well as the full λ -calculus. Since we focus here to only
 MLL, we can restrict to finite sets.

257 Notice that the cut composes the semantics of \mathbf{b}_1 with that of the proof-net containing \mathbf{b}_2
 258 and \mathbf{b}_3 , producing the sum over $y \in \llbracket X_1^+ \rrbracket^\iota$. Notice also that the contraction imposes that the
 259 same index y is shared between the two different boxes (\mathbf{b}_2 and \mathbf{b}_3): contraction duplicates
 260 the indexes of the vectors, but it does not yield different copies of the vectors themselves.
 261 This is in accordance with Remarks 4 and 6: sharing of sampled values corresponds here to
 262 sharing vector indices, which is different from duplicating whole vectors. If we consider in
 263 fact the proof-net $\llbracket \mathcal{N}'' \rrbracket^\iota$ given by a tensor between \mathbf{b}_2 and \mathbf{b}_3 and two *distinct* copies of \mathbf{b}_1 ,
 264 one cut with the X_1^- conclusion of \mathbf{b}_2 and the other one with that of \mathbf{b}_3 , then we would have:

$$265 \quad \llbracket \mathcal{N}'' \rrbracket_{(x', (x'', x''))}^\iota = \sum_{y, y' \in \llbracket X_1^+ \rrbracket^\iota} \iota(\mathbf{b}_1)_y \iota(\mathbf{b}_2)_{(y', x')} \iota(\mathbf{b}_1)_{y'} \iota(\mathbf{b}_3)_{(y', (x'', x''))}.$$

266 **► Example 9.** Let us consider a proof-net \mathcal{N} which is a bunch of $n + 1$ axioms over a tree of n
 267 contractions, of which edges are labelled by X^- , so that \mathcal{N} has conclusions X^-, X^+, \dots, X^+ .
 268 The denotation $\llbracket \mathcal{N} \rrbracket^\iota$ is then a vector indexed by the $(n + 2)$ -tuples of elements in $\iota(X)$.
 269 In fact, by using Figure 1c, one can check that $\llbracket \mathcal{N} \rrbracket^\iota$ is a very sparse vector, having zero
 270 everywhere but on the tuples of equal elements, i.e. (x, x, \dots, x) for $x \in \iota(X)$, in which
 271 case $\llbracket \mathcal{N} \rrbracket^\iota$ returns 1. We have here a first source of inefficiency of this kind of semantics,
 272 representing the denotation of a proof-net with a vector of dimension exponential in the
 273 number of its conclusions, where it would suffice a much more compact structure to store the
 274 same information. Section 5 will provide this structure with the notion of *component factor*.

275 If \mathcal{N} has several cuts, the computation of $\llbracket \mathcal{N} \rrbracket^\iota$ can be considerably simplified by using
 276 the following lemma, which is reminiscent of the notion of experiment introduced in [18].

277 **► Lemma 10 (Cut bundles).** *Let $\text{Cut}_\Gamma(\mathcal{N})$ be a proof-net of conclusions Δ that can be*
 278 *decomposed into a proof-net \mathcal{N} of conclusions $\Delta, \Gamma, \Gamma^\perp$ and a bundle of cuts between the*
 279 *formulas in Γ and Γ^\perp . Then, for every $\vec{d} \in \llbracket \Delta \rrbracket^\iota$, we have: $\llbracket \text{Cut}_\Gamma(\mathcal{N}) \rrbracket_{\vec{d}}^\iota = \sum_{\vec{c} \in \llbracket \Gamma \rrbracket^\iota} \llbracket \mathcal{N} \rrbracket_{(\vec{d}, \vec{c}, \vec{c})}^\iota$.*

280 **► Example 11.** Let us compute the semantics of the proof-net \mathcal{N}_0 in Figure 1e, by using
 281 Lemma 10 and Figure 1c. We have that, for any $(x_4, x_5) \in \llbracket X_4^+ \otimes X_5^+ \rrbracket^\iota$:

$$282 \quad \llbracket \mathcal{N}_0 \rrbracket_{(x_4, x_5)}^\iota = \sum_{\substack{x_i \in \iota(X_i^+) \\ \text{for } i \in \{1, 2, 3\}}} \iota(\mathbf{b}_1)_{x_1} \iota(\mathbf{b}_2)_{(x_1, x_2)} \iota(\mathbf{b}_3)_{(x_1, x_2, x_3)} \iota(\mathbf{b}_4)_{(x_2, x_3, x_4)} \iota(\mathbf{b}_5)_{(x_2, x_5)}$$

283 With a bit more of effort (due to the presence of axioms) also $\llbracket \mathcal{N} \rrbracket^\iota$ can be associated with
 284 the above summation. If we suppose that for every i , $\iota(X_i) = \{\mathbf{t}, \mathbf{f}\}$, this summation has a
 285 total of 2^3 terms, so that computing the whole vector $\llbracket \mathcal{N}_0 \rrbracket^\iota$ requires $\sim 2^5$ basic operations⁴,
 286 i.e. a quantity exponential in the number of the semantical boxes.

287 By carefully inspecting the summation, one can however realise that it can be refactored
 288 so to split factors over independent variables, getting for example the expression:

$$289 \quad \sum_{x_3} \left(\sum_{x_2} \left(\sum_{x_1} \iota(\mathbf{b}_1)_{x_1} \iota(\mathbf{b}_2)_{(x_1, x_2)} \iota(\mathbf{b}_3)_{(x_1, x_2, x_3)} \right) \iota(\mathbf{b}_4)_{(x_2, x_3, x_4)} \right) \iota(\mathbf{b}_5)_{(x_2, x_5)}$$

290 which, by memorising the intermediate sums, performs the same computation of $\llbracket \mathcal{N} \rrbracket^\iota$ in just
 291 $\sim 2^3$ operations. This kind of refactoring is at the core of many algorithms for exact inference
 292 in Bayesian graphs and the next sections will show how to import these methods.

⁴ We are supposing that multiplication, addition and coefficient access are operations of constant cost.

4 Factors

We adapt from Bayesian networks (e.g. [8]) the notion of factor (Definition 18) and of product and projection of factors. A factor carries both a vector *and* a “sharing structure” about what entries of this vector will be shared with possibly other factors so that we avoid the dimension explosion which is the source of inefficiency in Example 9. Bayesian networks use random variables for expressing such a “sharing structure”, while we reduce this latter into the very basic definition of set-family, which encompasses the former (Example 15) and generalises to whole quantitative MLL. The terminology “factor” is standard in Bayesian networks, in fact this notion refers to the terms in the multiplication giving a joint distribution as the outcome of the variable elimination algorithm (Algorithm 1). We introduce also a notion of renaming (Definition 29) and of factor renaming (Definition 34) necessary to follow the compositional structure of MLL (see discussion in Example 41).

► **Definition 12** (Set-family). *We call set-family a finite, indexed family of finite sets, i.e. a map \mathbb{X} from a finite set $\mathcal{I}(\mathbb{X})$ of indices to a set $\mathbf{Sets}(\mathbb{X})$ of finite sets. We denote by $\mathbb{X}(a)$ the set associated with index $a \in \mathcal{I}(\mathbb{X})$ in \mathbb{X} . Meta-variables $\mathbb{X}, \mathbb{Y}, \mathbb{Z}$ will range over such set-families.*

Two families \mathbb{X} and \mathbb{Y} are compatible whenever for all $a \in \mathcal{I}(\mathbb{X}) \cap \mathcal{I}(\mathbb{Y})$, $\mathbb{X}(a) = \mathbb{Y}(a)$. Set-theoretical operations lift to compatible set-families by applying the former to the graph of these latter, e.g. the intersection $\mathbb{X} \cap \mathbb{Y}$ is the set-family defined by $\mathcal{I}(\mathbb{X} \cap \mathbb{Y}) ::= \mathcal{I}(\mathbb{X}) \cap \mathcal{I}(\mathbb{Y})$ and $(\mathbb{X} \cap \mathbb{Y})(a) ::= \mathbb{X}(a) = \mathbb{Y}(a)$ for every $a \in \mathcal{I}(\mathbb{X} \cap \mathbb{Y})$. Similarly, we will consider the union $\mathbb{X} \cup \mathbb{Y}$ and the set-theoretical difference $\mathbb{X} \setminus \mathbb{Y}$. In the same spirit, we write $\mathbb{Y} \subseteq \mathbb{X}$, for $\mathcal{I}(\mathbb{Y}) \subseteq \mathcal{I}(\mathbb{X})$ and for every $a \in \mathcal{I}(\mathbb{Y})$, $\mathbb{Y}(a) = \mathbb{X}(a)$.

Given a set-family \mathbb{X} , we denote by $\llbracket \mathbb{X} \rrbracket$ the cartesian product $\prod_{a \in \mathcal{I}(\mathbb{X})} \mathbb{X}(a)$ of the sets in $\mathbf{Sets}(\mathbb{X})$, where the same set in $\mathbf{Sets}(\mathbb{X})$ can appear multiple times in the product if associated with multiple indices. We denote the elements of $\llbracket \mathbb{X} \rrbracket$ with the vectorial notation \vec{x} , to underline that it is an element in a cartesian product rather than in a simple set.

► **Notation 13.** *Any element $\vec{x} \in \llbracket \mathbb{X} \rrbracket$ can be seen as a collection $(x_a)_{a \in \mathcal{I}(\mathbb{X})}$ of elements in $\mathbf{Sets}(\mathbb{X})$. In particular, given $\mathbb{Y} \subseteq \mathbb{X}$, we denote by $\vec{x}|_{\mathbb{Y}}$ the projected element $(x_a)_{a \in \mathcal{I}(\mathbb{Y})} \in \llbracket \mathbb{Y} \rrbracket$. Similarly, given two set-families \mathbb{X}, \mathbb{Y} having disjoint sets of indexes, so clearly compatible, the elements of $\llbracket \mathbb{X} \uplus \mathbb{Y} \rrbracket$ can be written as (\vec{x}, \vec{y}) , for $\vec{x} \in \llbracket \mathbb{X} \rrbracket$ and $\vec{y} \in \llbracket \mathbb{Y} \rrbracket$.*

Notice that if \mathbb{X} is empty, then $\llbracket \mathbb{X} \rrbracket$ is the singleton set $\{()\}$.

► **Notation 14.** *Since finite, set-families can be given by enumerating their graph, like in $\mathbb{X} = \{(a_1, S_1), \dots, (a_n, S_n)\}$. In this case we have: $\mathcal{I}(\mathbb{X}) = \{a_1, \dots, a_n\}$ and $\mathbf{Sets}(\mathbb{X}) = \{S_1, \dots, S_n\}$. In this latter set, the possible repetitions are equated, so $\mathbf{Sets}(\mathbb{X})$ might have less than n elements.*

► **Example 15.** A finite set $\{X_1, \dots, X_n\}$ of finite random variables defines the set-family $\mathbb{X} = \{(X_1, \llbracket X_1 \rrbracket), \dots, (X_n, \llbracket X_n \rrbracket)\}$, where $\llbracket X_i \rrbracket$ denotes the finite set of the possible outcomes taken by the random variable X_i . Notice that $\llbracket \mathbb{X} \rrbracket$ is then the set of samples of the joint distribution over X_1, \dots, X_n . To be more explicit, suppose that each random variable X_i is boolean, i.e. $\llbracket X_i \rrbracket = \{\mathbf{t}, \mathbf{f}\}$ for all $i \leq n$, then $\mathbf{Sets}(\mathbb{X}) = \{\{\mathbf{t}, \mathbf{f}\}\}$, while $\llbracket \mathbb{X} \rrbracket = \{(b_1, \dots, b_n) \mid b_i \in \{\mathbf{t}, \mathbf{f}\}\}$.

► **Example 16.** Consider a sequent $\Gamma = X_1^\circ, \dots, X_n^\circ$ of atomic formulas. A natural set-family that can be associated with Γ and a valuation ι , has indices the sequent positions $\{1, \dots, n\}$ and it maps a position i to the set $\iota(X_i)$. This set-family however is not the only possible one: for example, one may take as indices the propositional variables X_1, \dots, X_n , where

4:10 Sum-Product for MLL

multiple occurrences of the same variable are equated, and map X_i to $\iota(X_i)$. The two set-families are quite different if Γ contains repetitions. Namely, let $\Gamma = X^+, X^+, X^-, Y^-$, with $\llbracket X \rrbracket = \llbracket Y \rrbracket = \{\mathbf{t}, \mathbf{f}\}$. The two set-families are:

$$\mathbb{X} = \{(1, \{\mathbf{t}, \mathbf{f}\}), (2, \{\mathbf{t}, \mathbf{f}\}), (3, \{\mathbf{t}, \mathbf{f}\}), (4, \{\mathbf{t}, \mathbf{f}\})\}, \quad \mathbb{Y} = \{(X, \{\mathbf{t}, \mathbf{f}\}), (Y, \{\mathbf{t}, \mathbf{f}\})\}.$$

► **Remark 17.** Notice that $\mathbb{Z}, \mathbb{Y} \subseteq \mathbb{X}$ implies that both \mathbb{Z} and \mathbb{Y} are compatible. Henceforth we will always consider families which are subset of a fixed “universal” family (underlined by a proof-net), so that the compatibility condition in Definition 12 is not an issue and hence will be often not mentioned.

► **Definition 18 (Factor).** A generalised factor, or simply factor, ϕ is a pair $(\text{Fam}(\phi), \text{Fun}(\phi))$ of a set-family $\text{Fam}(\phi)$ and a function $\text{Fun}(\phi)$ from the set $\llbracket \text{Fam}(\phi) \rrbracket$ to $\mathbb{R}_{\geq 0}$.

We will short the notation $\text{Fun}(\phi)$ by writing just ϕ when it is clear from the context that we are considering the function associated with a factor and not the whole pair $(\text{Fam}(\phi), \text{Fun}(\phi))$. We often consider $\text{Fun}(\phi)$ as a vector indexed by the elements of its domain, so that $\phi_{\vec{x}}$ stands for $\text{Fun}(\phi)(\vec{x})$, for every $\vec{x} \in \llbracket \text{Fam}(\phi) \rrbracket$.

► **Example 19.** Let us recall the set-family $\mathbb{Y} = \{(X, \{\mathbf{t}, \mathbf{f}\}), (Y, \{\mathbf{t}, \mathbf{f}\})\}$ of Example 16, and consider the function $\text{Fun}(\phi)$ given by $\{(\mathbf{t}_X, \mathbf{t}_Y) \mapsto 0.2, (\mathbf{t}_X, \mathbf{f}_Y) \mapsto 0.25, (\mathbf{f}_X, \mathbf{t}_Y) \mapsto 0.25, (\mathbf{f}_X, \mathbf{f}_Y) \mapsto 0.3\}$. The pair $\phi = (\mathbb{Y}, \text{Fun}(\phi))$ is an example of factor. Intuitively, ϕ can be seen as the presentation $0.2e_{(\mathbf{t}_X, \mathbf{t}_Y)} + 0.25e_{(\mathbf{t}_X, \mathbf{f}_Y)} + 0.25e_{(\mathbf{f}_X, \mathbf{t}_Y)} + 0.3e_{(\mathbf{f}_X, \mathbf{f}_Y)}$ of a vector in $\mathbb{R}_{\geq 0}^4$ with respect to a set of basis vectors $e_{(b_X, b_Y)}$ associated with the elements in $(b_X, b_Y) \in \llbracket \mathbb{Y} \rrbracket$.

► **Definition 20 (Factor projection).** Let ϕ be a factor and let \mathbb{X} be a set-family compatible with $\text{Fam}(\phi)$, the projection of ϕ to \mathbb{X} is the factor $\pi_{\mathbb{X}}(\phi)$ defined by:

$$\text{Fam}(\pi_{\mathbb{X}}(\phi)) ::= \mathbb{X}, \quad \pi_{\mathbb{X}}(\phi)_{\vec{x}} ::= \sum_{\vec{y} \in \llbracket \text{Fam}(\phi) \rrbracket \setminus \mathbb{X}} \phi_{\vec{x} \upharpoonright_{\text{Fam}(\phi)}, \vec{y}}, \quad \text{for } \vec{x} \in \llbracket \mathbb{X} \rrbracket.$$

► **Example 21.** Recall the set-family \mathbb{Y} and the factor $\text{Fun}(\phi)$ given in Example 19, let $\mathbb{X} = \{(X, \{\mathbf{t}, \mathbf{f}\})\} \subseteq \mathbb{Y}$. We have that $\pi_{\mathbb{X}}(\phi) = \{\mathbf{t}_X \mapsto 0.45, \mathbf{f}_X \mapsto 0.55\}$. Let now $\mathbb{Z} = \mathbb{X} \uplus \{(Z, \{\mathbf{t}, \mathbf{f}\})\}$, we have that $\pi_{\mathbb{Z}}(\phi) = \{(\mathbf{t}_X, \mathbf{t}_Z) \mapsto 0.45, (\mathbf{t}_X, \mathbf{f}_Z) \mapsto 0.45, (\mathbf{f}_X, \mathbf{t}_Z) \mapsto 0.55, (\mathbf{f}_X, \mathbf{f}_Z) \mapsto 0.55\}$. Notice in particular that the factor projection to a set-family \mathbb{Z} does not preserve in general the property of being a probability mass function, unless $\mathbb{Z} \subseteq \text{Fam}(\phi)$.

► **Remark 22.** With the notations of Definition 20, if $\mathbb{X} \subseteq \text{Fam}(\phi)$, then $\pi_{\mathbb{X}}(\phi)$ corresponds to what is called in Bayesian programming *summing out* $\text{Fam}(\phi) \setminus \mathbb{X}$, which gives the marginal distribution over \mathbb{X} . Suppose on the contrary that \mathbb{X} and $\text{Fam}(\phi)$ are disjoint, then for every $\vec{x} \in \llbracket \mathbb{X} \rrbracket$, $\pi_{\mathbb{X}}(\phi)_{\vec{x}}$ is the total mass of ϕ , i.e. $\sum_{\vec{y} \in \llbracket \text{Fam}(\phi) \rrbracket} \phi_{\vec{y}}$.

► **Remark 23.** Suppose that $\text{Fam}(\phi)$ has n indices and that k is the maximum cardinality of a set in $\text{Sets}(\text{Fam}(\phi))$, then the computation of the whole vector $\pi_{\mathbb{X}}(\phi)$ is in $O(k^n)$.

► **Definition 24 (Binary factor product).** Given two factors ϕ and ψ , such that $\text{Fam}(\phi)$ and $\text{Fam}(\psi)$ are compatible, we define their factor product as the factor $\phi \odot \psi$ given by:

$$\text{Fam}(\phi \odot \psi) ::= \text{Fam}(\phi) \cup \text{Fam}(\psi), \quad (\phi \odot \psi)_{\vec{z}} ::= \phi_{\vec{z} \upharpoonright_{\text{Fam}(\phi)}} \psi_{\vec{z} \upharpoonright_{\text{Fam}(\psi)}}, \quad \text{for } \vec{z} \in \llbracket \text{Fam}(\phi \odot \psi) \rrbracket.$$

► **Remark 25.** If $\text{Fam}(\phi) \cup \text{Fam}(\psi)$ has n indices and k is the maximum cardinality of a set in $\text{Sets}(\text{Fam}(\phi) \cup \text{Fam}(\psi))$, then the computation of the whole vector $\phi \odot \psi$ is in $O(k^n)$.

380 ► **Example 26.** In terms of MLL operations, factor products correspond to a \otimes product plus
 381 a bunch of contractions on the common indexes. For example, let us take as indexes the
 382 propositional variables and as sets just $\{\mathbf{t}, \mathbf{f}\}$ (recall Example 16) and consider $\mathbf{Fam}(\phi) =$
 383 $\{X_2, X_3, X_4\}$ and $\mathbf{Fam}(\psi) = \{X_2, X_5\}$ (this choice is reminiscent of the variables in the
 384 proof-nets in Figure 1e, in fact ϕ and ψ can be associated with the boxes, respectively, \mathbf{b}_4 and
 385 \mathbf{b}_5). Then, $\mathbf{Fun}(\phi \odot \psi)$ is over $\{X_2, X_3, X_4, X_5\}$, so of dimension 2^4 , while $\mathbf{Fun}(\phi) \otimes \mathbf{Fun}(\psi)$
 386 is a vector indexed by tuples of 5 booleans, so of dimension 2^5 .

387 The next proposition states expected properties of factor projection and product that are
 388 fundamental in the sequel.

389 ► **Proposition 27.** *Factor product is associative and commutative, with neutral element the*
 390 *empty factor $(\emptyset, 1)$. Moreover:*

- 391 1. $\pi_{\mathbb{X} \cup \mathbb{Z}}(\pi_{\mathbb{X} \cup \mathbb{Y}}(\phi)) = \pi_{\mathbb{X} \cup \mathbb{Z}}(\phi)$, whenever $\mathbb{Y} \subseteq \mathbf{Fam}(\phi)$ and $\mathbb{Z} \cap \mathbf{Fam}(\phi) = \emptyset$;
- 392 2. $\pi_{\mathbb{X}}(\phi \odot \psi) = \pi_{\mathbb{X}}(\phi) \odot \psi$, whenever $\mathbf{Fam}(\psi) \subseteq \mathbb{X}$.

393 ► **Definition 28** (*n-factor product*). *Let I be a finite set. Given a collection of pairwise*
 394 *compatible factors $(\phi_i)_{i \in I}$, we define their factor product as the factor $\bigodot_{i \in I} \phi_i ::= \phi_{i_1} \odot$*
 395 *$\cdots \odot \phi_{i_n}$, for some enumeration of I . This is well-defined independently from the chosen*
 396 *enumeration because of Proposition 27.*

397 Section 5 will associate factors to MLL proof-nets and in order to make this association
 398 compositional (Theorem 48) we introduce the following notion of renaming, as the contraction
 399 and cut rules of Figure 1b may change the sharing structure associated with a proof-net.

400 ► **Definition 29** (Renaming). *A renaming f from a set-family \mathbb{X} to a set-family \mathbb{Y} is a map*
 401 *from $\mathcal{I}(\mathbb{X})$ to $\mathcal{I}(\mathbb{Y})$ such that for all $a \in \mathcal{I}(\mathbb{X})$, we have $\mathbb{X}(a) = \mathbb{Y}(f(a))$. Any such renaming*
 402 *f induces the map f° from $[[\mathbb{Y}]]$ to $[[\mathbb{X}]]$ by:*

$$403 \quad \text{for } \vec{y} \in [[\mathbb{Y}]], \quad f^\circ(\vec{y}) ::= (y_{f(a)})_{a \in \mathcal{I}(\mathbb{X})} \in [[\mathbb{X}]]. \quad (2)$$

405 *Moreover, we say that a point $\vec{x} \in [[\mathbb{X}]]$ agrees on f whenever, for every $a, a' \in \mathcal{I}(\mathbb{X})$,*
 406 *$f(a) = f(a')$ implies that $x_a = x_{a'}$.*

407 ► **Remark 30.** The notion of “agreeing on a renaming f ” generalises the notion in Bayesian
 408 programming of a set of samples that “agrees on the same random variables” as used in
 409 e.g. [8].

410 ► **Notation 31.** *Given a renaming f from \mathbb{X} to \mathbb{Y} , and a set-family $\mathbb{X}' \subseteq \mathbb{X}$, we denote by*
 411 *$f(\mathbb{X}')$ the set-family having as indices the set $f(\mathcal{I}(\mathbb{X}')) \subseteq \mathcal{I}(\mathbb{Y})$ and that it associates with*
 412 *any $b \in f(\mathcal{I}(\mathbb{X}'))$ the set $\mathbb{Y}(b)$. Notice that $f(\mathbb{X}') \subseteq \mathbb{Y}$.*

413 ► **Proposition 32.** *Given a renaming from \mathbb{X} to \mathbb{Y} , the image set of f° is the subset of $[[\mathbb{X}]]$*
 414 *of the elements which agree on f . If, moreover, f is surjective over $\mathcal{I}(\mathbb{Y})$, then f° is an*
 415 *injective map from $[[\mathbb{Y}]]$ to $[[\mathbb{X}]]$, hence a bijection from $[[\mathbb{Y}]]$ to $\{\vec{x} \in [[\mathbb{X}]] \mid \vec{x} \text{ agrees on } f\}$. In*
 416 *this case, we denote its inversion by f^\bullet .*

417 ► **Example 33.** Recall the set-families \mathbb{X} and \mathbb{Y} in Example 16, associated with the sequent
 418 $\Gamma = X^+, X^+, X^-, Y^-$. Let us consider the following two specific renamings from \mathbb{X} to \mathbb{Y} :

$$419 \quad f = \begin{cases} 1, 2, 3 & \mapsto X \\ 4 & \mapsto Y \end{cases}, \quad g = \begin{cases} 1, 2, 3, 4 & \mapsto X \end{cases}.$$

4:12 Sum-Product for MLL

and take for example $\vec{y} = (\mathfrak{t}_X, \mathfrak{f}_Y) \in \llbracket \mathbb{Y} \rrbracket$. We have (we use the natural order (1,2,3,4) to represent elements in $\llbracket \mathbb{X} \rrbracket$):

$$f^\circ(\vec{y}) = (\mathfrak{t}, \mathfrak{t}, \mathfrak{t}, \mathfrak{f}), \quad g^\circ(\vec{y}) = (\mathfrak{t}, \mathfrak{t}, \mathfrak{t}, \mathfrak{t}).$$

Notice in fact that $f^\circ(\vec{y})$ agrees on f but not on g , while $g^\circ(\vec{y})$ agrees on both f and g . Also notice that f is surjective and in fact f° is an injection, while g is not surjective and in fact g° is not a injection, for example: $g^\circ(\mathfrak{t}_X, \mathfrak{f}_Y) = g^\circ(\mathfrak{t}_X, \mathfrak{t}_Y)$.

► **Definition 34 (Factor renaming).** Let f be a renaming from $\text{Fam}(\phi)$ to a set-family \mathbb{X} . The renaming of ϕ along f is the factor $f(\phi)$ defined by:

$$\text{Fam}(f(\phi)) ::= f(\text{Fam}(\phi)), \quad f(\phi)_{\vec{x}} ::= \phi_{f^*(\vec{x})}, \quad \text{for } \vec{x} \in \llbracket f(\text{Fam}(\phi)) \rrbracket.$$

► **Example 35.** Recall the renaming f of Example 33 between the set-families \mathbb{X} and \mathbb{Y} given in Example 16. Consider the factor ϕ over \mathbb{X} defined by $\phi_{(b_1, b_2, b_3, b_4)} ::= 1$ if $b_1 = b_2 = b_3$, 0 otherwise. This factor corresponds to the interpretation of the proof-net having a weakening producing Y^- and the axiom on top of a contraction giving X^+, X^+, X^- . Then $f(\phi)$ is over \mathbb{Y} and its map is the constant function giving 1.

► **Remark 36.** Notice that one can formalise the notions of this section in a categorical way, considering a category of renamings as morphisms between set-families. We did not develop this more abstract presentation as not needed in the sequel.

5 Weighted Semantics by Factors, Atomic Case

We apply to MLL the notions introduced in Section 4. It is convenient to restrict to atomic proof-nets and then to extend the results to the non-atomic case in Section 6. Definitions 37 and 45 associate two different set-families with an atomic proof-net \mathcal{N} , the edge and the component set-families. The edge set-family permits to consider the standard weighted interpretation $\llbracket \mathcal{N} \rrbracket$ as a factor (Definition 39), while the component set-family yields a more compressed representation of $\llbracket \mathcal{N} \rrbracket$, the component factor (Definition 40), which has a form of compositionality (Theorem 48) and hence can be computed directly (Theorem 49) without using the rules of Figure 1c. Henceforth, this section fixes a valuation ι and considers only atomic proof-nets. We recall that an atomic proof-net can contain only axioms, cuts, weakening, contractions and semantical boxes.

► **Definition 37 (Edge set-family).** Let \mathcal{N} be an atomic proof-net and ι be a valuation. The edge set-family of \mathcal{N} , written by $\text{Fam}_e^\iota(\mathcal{N})$, has the edges of \mathcal{N} as indices and associates with an edge $e : X^\circ$ the set $\iota(X)$. Given a sequence Γ of edges $e_1 : X_1^\circ, \dots, e_n : X_n^\circ$, we extend the metavariable Γ to denote also the edge set-family $\{(e_1, \iota(X_1)), \dots, (e_n, \iota(X_n))\} \subseteq \text{Fam}_e^\iota(\mathcal{N})$.

► **Remark 38.** Let Γ be $e_1 : X_1^\circ, \dots, e_n : X_n^\circ$. The convention of denoting by Γ both the underlined sequent $X_1^\circ, \dots, X_n^\circ$ and the edge set-family $\{(e_1, \iota(X_1)), \dots, (e_n, \iota(X_n))\}$ is coherent because the cartesian product $\llbracket \Gamma \rrbracket$ associated with the edge set-family is the same set as the weighted denotation $\llbracket \Gamma \rrbracket^\iota$, this latter also denoted simply by $\llbracket \Gamma \rrbracket$. This ease of notation is necessary to avoid a formalism overkill.

► **Definition 39.** Given a valuation ι , the edge factor of an atomic proof-net \mathcal{N} has as set-family the edge-set family induced by the conclusions of \mathcal{N} and as function the weighted interpretation $\llbracket \mathcal{N} \rrbracket^\iota$. We take the liberty to denote this edge factor also by $\llbracket \mathcal{N} \rrbracket^\iota$.

463 ► **Definition 40** (Component set-family and renaming). Let \mathcal{N}^{-b} denote the graph obtained
 464 from an atomic proof-net \mathcal{N} by removing all its semantical boxes, so keeping their conclusions
 465 as pending edges of the graph. Given an undirected connected component \mathcal{M} of \mathcal{N}^{-b} , one can
 466 remark that all edges of \mathcal{M} are atomic formulas over a unique variable, let us denote it $X_{\mathcal{M}}$.
 467 The component set-family of \mathcal{N} , written $\text{Fam}_c^t(\mathcal{N})$, has as indices the undirected connected
 468 components of \mathcal{N}^{-b} and associates with a component \mathcal{M} the set $\iota(X_{\mathcal{M}})$. The component
 469 renaming of \mathcal{N} , written $\ell_{\mathcal{N}}$, is the renaming from $\text{Fam}_e^t(\mathcal{N})$ to $\text{Fam}_c^t(\mathcal{N})$ mapping the edges
 470 of \mathcal{N} to the connected component of \mathcal{N}^{-b} they belong to.

471 ► **Example 41.** Consider the atomic proof-net \mathcal{N}_a obtained from the proof-net \mathcal{N}_0 in Figure 1e
 472 by removing the tensor node, so that \mathcal{N}_a has conclusions X_4^+, X_5^+ . Notice that \mathcal{N}_a^{-b} has
 473 five connected components which correspond to the five propositional variables X_1, \dots, X_5 .
 474 If however we consider the sub proof-net \mathcal{N}'_a obtained from \mathcal{N}_a by removing the cut and
 475 the contraction insisting on the edges typed by, e.g., X_1^- , we have that \mathcal{N}'_a^{-b} has now seven
 476 connected components, in particular three of them supports the same variable X_1 . This
 477 example shows that the generating rules of MLL (Figure 1b) require not to mix up the
 478 component set-family with the variables labelling the edges (see also Remark 55).

479 ► **Remark 42.** The $\ell_{\mathcal{N}}$ information can be memorised once and for all by adding a further
 480 labelling over the edges of \mathcal{N} giving the same index to the edges belonging to the same
 481 connected component of \mathcal{N}^{-b} . This labelling can be computed in linear time with respect to
 482 the size of \mathcal{N} , by adapting one of the many connected component algorithms.

483 ► **Notation 43.** Let $\Gamma = e_1 : X_1^\circ, \dots, e_n : X_n^\circ$ be a set of edges of \mathcal{N} , so that Γ is also the
 484 set-family $\{(e_1, \iota(X_1)), \dots, (e_n, \iota(X_n))\}$ contained in $\text{Fam}_e^t(\mathcal{N})$. By Notation 31, the writing
 485 $\ell_{\mathcal{N}}(\Gamma)$ denotes the set-family $\{(\ell_{\mathcal{N}}(e_1), \iota(X_1)), \dots, (\ell_{\mathcal{N}}(e_n), \iota(X_n))\} \subseteq \text{Fam}_c^t(\mathcal{N})$. Notice that
 486 $\ell_{\mathcal{N}}$ is surjective on $\mathcal{I}(\ell_{\mathcal{N}}(\Gamma))$, so we can apply Proposition 32, getting the two maps:
 487 ■ $\ell_{\mathcal{N}}^\circ$ from $\llbracket \Gamma \rrbracket$ to $\llbracket \ell_{\mathcal{N}}(\Gamma) \rrbracket$,
 488 ■ its inverse $\ell_{\mathcal{N}}^\bullet$ from $\{\vec{d} \in \llbracket \Gamma \rrbracket \mid \vec{d} \text{ agrees on } \ell_{\mathcal{N}}\}$ to $\llbracket \ell_{\mathcal{N}}(\Gamma) \rrbracket$.

489 ► **Remark 44.** In general, given a set of edges Γ of an atomic proof-net \mathcal{N} , we have that:
 490 $\llbracket \ell_{\mathcal{N}}(\Gamma) \rrbracket = \llbracket \Gamma \rrbracket$ if, and only if, all edges in Γ belong to pairwise different connected components
 491 of \mathcal{N}^{-b} . Moreover, if $\llbracket \ell_{\mathcal{N}}(\Gamma) \rrbracket = \llbracket \Gamma \rrbracket$, then every $\vec{d} \in \llbracket \Gamma \rrbracket$ agrees on $\ell_{\mathcal{N}}$.

492 ► **Definition 45.** Given a valuation ι , the component factor of an atomic proof-net \mathcal{N} is the
 493 renaming $\ell_{\mathcal{N}}(\llbracket \mathcal{N} \rrbracket^\iota)$ of its edge factor, i.e. if Δ are the conclusions of \mathcal{N} , $\text{Fam}(\ell_{\mathcal{N}}(\llbracket \mathcal{N} \rrbracket^\iota)) =$
 494 $\ell_{\mathcal{N}}(\Delta)$ and for $\vec{d} \in \llbracket \ell_{\mathcal{N}}(\Delta) \rrbracket$, $\text{Fun}(\ell_{\mathcal{N}}(\llbracket \mathcal{N} \rrbracket^\iota))_{\vec{d}} = \llbracket \mathcal{N} \rrbracket_{\ell_{\mathcal{N}}^\circ(\vec{d})}^\iota$.

495 ► **Example 46.** Recall the proof-net \mathcal{N} and the valuation ι of Example 9. Notice that
 496 $\text{Fam}(\ell_{\mathcal{N}}(\llbracket \mathcal{N} \rrbracket^\iota))$ is a singleton as the $n + 2$ conclusions of \mathcal{N} belong to the same component,
 497 and $\text{Fun}(\ell_{\mathcal{N}}(\llbracket \mathcal{N} \rrbracket^\iota)) = (1_{\mathfrak{t}}, 1_{\mathfrak{f}})$, which is a more parsimonious object than $\llbracket \mathcal{N} \rrbracket^\iota$, this latter of
 498 dimension exponential in n .

499 Proposition 47 details how to recover the original denotation of \mathcal{N} out of its component
 500 factor.

501 ► **Proposition 47.** Let \mathcal{N} be an atomic proof-net of conclusions Δ . For every $\vec{d} \in \llbracket \Delta \rrbracket$, we
 502 have that:

$$503 \quad \llbracket \mathcal{N} \rrbracket_{\vec{d}}^\iota = \begin{cases} \ell_{\mathcal{N}}(\llbracket \mathcal{N} \rrbracket^\iota)_{\ell_{\mathcal{N}}^\bullet(\vec{d})} & \text{if } \vec{d} \text{ agrees on } \ell_{\mathcal{N}}, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

504 The following is the core theorem of this paper: MLL cuts correspond to a factor product
505 plus a projection.

506 ► **Theorem 48.** *Let $\mathcal{N} = \text{Cut}_\Gamma(\mathcal{N}', \mathcal{N}'')$ be an atomic proof-net of conclusions Δ obtained
507 by connecting by a bunch of cuts over a sequent Γ a sub proof-net \mathcal{N}' , of conclusions Γ, Δ' ,
508 and \mathcal{N}'' , of conclusions Γ^\perp, Δ'' , so that $\Delta = \Delta', \Delta''$. We have:*

$$509 \quad \ell_{\mathcal{N}}(\llbracket \mathcal{N} \rrbracket^\iota) = \pi_{\ell_{\mathcal{N}}(\Delta)}(\ell_{\mathcal{N}}(\llbracket \mathcal{N}' \rrbracket^\iota) \odot \ell_{\mathcal{N}}(\llbracket \mathcal{N}'' \rrbracket^\iota)) \quad (4)$$

510 As a consequence, we can compute the component factor of \mathcal{N} without passing via $\llbracket \mathcal{N} \rrbracket^\iota$:

511 ► **Theorem 49.** *Let \mathcal{N} be an atomic proof-net with conclusions Δ . We have: $\ell_{\mathcal{N}}(\llbracket \mathcal{N} \rrbracket^\iota) =$
512 $\pi_{\ell_{\mathcal{N}}(\Delta)}(\bigodot_{\mathbf{b} \in \mathbf{b}(\mathcal{N})} \ell_{\mathcal{N}}(\iota(\mathbf{b})))$.*

513 ► **Example 50.** Consider an atomic proof-net \mathcal{N} of conclusions Δ such that no edge in Δ is
514 connected in $\mathcal{N}^{-\mathbf{b}}$ with a conclusion of a box of \mathcal{N} . By Remark 22, $\pi_{\ell_{\mathcal{N}}(\Delta)}(\bigodot_{\mathbf{b} \in \mathbf{b}(\mathcal{N})} \iota(\mathbf{b}))$ is
515 the constant function giving the total mass of the vectors associated with the boxes of \mathcal{N} .

516 Consider the atomic proof-net \mathcal{N}_a obtained by removing the tensor node from the proof-net
517 \mathcal{N}_0 of Figure 1e. If we apply Theorem 49 to \mathcal{N}_a , we will obtain exactly the same summation
518 given in Example 11, in fact the edge and component set-families of the conclusions of
519 \mathcal{N}_a are the same. However, we have now the correct formalism to apply exact inference
520 algorithms to refactor the expression $\pi_{\ell_{\mathcal{N}}(\Delta)}(\bigodot_{\mathbf{b} \in \mathbf{b}(\mathcal{N})} \ell_{\mathcal{N}}(\iota(\mathbf{b})))$, by taking advantage of the
521 distributivity law of factor product over the projection (Proposition 27). We adapt here one
522 among the simplest such algorithms, called the sum-product variable elimination algorithm,
523 first introduced in [29], see [8] as a reference. The terminology “variable elimination” is
524 because this procedure infers from a Bayesian network the marginal distribution of a random
525 variable X out of a family⁵ \mathbb{X} of variables containing X , by “eliminating” all the other
526 variables in \mathbb{X} . In our case, what we “eliminate” are the $\mathcal{N}^{-\mathbf{b}}$ components of the conclusions
527 of the box factors containing no conclusion of the proof-net.

■ Algorithm 1 MLL Sum-Product Algorithm

input:

- 1: \mathcal{N} ▷ an atomic proof-net of conclusions Δ
- 2: ι ▷ a valuation map
- 3: ω ▷ A linear order on the components in $\text{Fam}_c^t(\mathcal{N})$ not having a conclusion in Δ

output: the factor $\ell_{\mathcal{N}}(\llbracket \mathcal{N} \rrbracket^\iota)$

- 4: $F \leftarrow \{\ell_{\mathcal{N}}(\iota(\mathbf{b})) \mid \mathbf{b} \in \mathbf{b}(\mathcal{N})\}$ ▷ Factors of $\mathbf{b}(\mathcal{N})$
- 5: **for** C in ω **do**
- 6: $F_c \leftarrow \{\phi \in F \mid C \in \mathcal{I}(\text{Fam}(\phi))\}$
- 7: $\psi \leftarrow \bigodot_{\phi \in F_c} \phi$ ▷ Product
- 8: $\rho \leftarrow \pi_{\text{Fam}(\psi) \setminus \{C\}}(\psi)$ ▷ Sum-out
- 9: $F \leftarrow \{\rho\} \cup (F \setminus F_c)$
- return** $\bigodot_{\phi \in F} \phi$

528 Algorithm 1 is our adaptation of the sum-product algorithm. Given a linear order ω
529 over the connected components of $\mathcal{N}^{-\mathbf{b}}$ which contains no conclusion of \mathcal{N} , the algorithm
530 proceeds as follows: line 4 initialises a variable F with the set of factors to compute; line 5

⁵ Any resemblance to the notations in Section 4 is purely voluntary.

531 takes from ω the next connected component C to process; line 6 gathers in a variable F_c all
 532 factors in F which have C as an index (i.e. a conclusion in C); line 7 computes the product
 533 of these factors and line 8 projects it on the components different from C (a.k.a. summing
 534 out C); line 9 updates F by replacing the processed factors with the result of this projection
 535 and then it jumps back to line 5. At the end of this loop, F contains a set of factors indexed
 536 over the components of \mathcal{N}^{-b} connected with the conclusions in \mathcal{N} and then it returns their
 537 product.

538 Soundness follows from Proposition 27 and Theorem 49:

539 ► **Theorem 51.** *Algorithm 1 returns $\ell_{\mathcal{N}}(\llbracket \mathcal{N} \rrbracket^{\iota})$ if fed with an atomic proof-net \mathcal{N} , a valuation*
 540 *ι and a linear order on the components in $\text{Fam}_c^{\iota}(\mathcal{N})$ not containing any conclusion of \mathcal{N} .*

541 ► **Example 52.** Consider the atomic proof-net \mathcal{N}_a obtained by removing the tensor node
 542 from the proof-net \mathcal{N}_0 of Figure 1e (Example 1) and use the numbers 1, 2, 3, 4, 5 to denote
 543 the five connected components of \mathcal{N}_a^{-b} such that component i is supported by variable X_i .
 544 The components to eliminate are 1, 2, 3. By taking the order $\omega = 1 < 2 < 3$, Algorithm 1
 545 will calculate the following intermediate factors: $\rho_1 = \pi_{\{2,3\}}(\iota(\mathbf{b}_1) \odot \iota(\mathbf{b}_2) \odot \iota(\mathbf{b}_3))$, $\rho_2 =$
 546 $\pi_{\{2,4\}}(\rho_1 \odot \iota(\mathbf{b}_4))$, $\rho_3 = \pi_{\{4,5\}}(\rho_2 \odot \iota(\mathbf{b}_5))$, the output being ρ_3 . This yields exactly the
 547 factored equation in Example 11 and it allows to calculate the whole semantics of \mathcal{N}_a
 548 in $O(k^3)$ basic operations, if k is the maximal cardinality of the sets associated with the
 549 propositional variables appearing in the proof-net.

550 ► **Remark 53.** A run of Algorithm 1 depends on the chosen order ω . Different orders yield
 551 different factorisations and have different performances. For example, by taking the inverse
 552 order $3 < 2 < 1$ in Example 52 we get a run in $O(k^4)$, which is an order of magnitude slower
 553 than $1 < 2 < 3$, although yet more efficient than the immediate recursive algorithm induced
 554 by the standard semantics (Example 11).

555 In general, Algorithm 1 is in $O(nk^w)$, where n is the length of ω (i.e. the number of
 556 components to eliminate), k is the maximal cardinality of a set interpreting an atomic variable
 557 (in our examples we always suppose $k = 2$, for the two booleans) and w is the maximal
 558 cardinality of $\text{Fam}(\phi)$, for ϕ a factor created/used by the algorithm (this parameter depends
 559 on the chosen order ω).

560 The quest for optimal orders is a major topic in Bayesian networks, which is however
 561 known to be a NP-hard problem [4]. Since probabilistic MLL contain Bayesian networks, we
 562 should focus on heuristics that yield good performances in most cases.

563 ► **Remark 54.** Recall the proof-net \mathcal{N} in Figure 1e which cut reduces to \mathcal{N}_0 . Notice that
 564 this proof-net does not resemble to a Bayesian network, e.g. it alternates par and tensor
 565 nodes. However, the reader may recognise the intermediate factors ρ_1 , ρ_2 and ρ_3 computed
 566 by Algorithm 1 in Example 52 as the nested sub proof-nets of \mathcal{N} of conclusions, respectively,
 567 $X_2^+ \otimes (X_2^- \wp X_3^+)$, $X_2^+ \otimes X_4^+$ and $X_4^+ \otimes X_5^+$. This is far from being a coincidence, as any run
 568 of Algorithm 1 can in fact be associated with a MLL proof-net, although this latter might
 569 need formulas with an arbitrary number of alternations between tensors and pars. We will
 570 investigate this point in a forthcoming paper.

571 ► **Remark 55.** The component renaming $\ell_{\mathcal{N}}$ is omitted in the setting of Bayesian networks
 572 as encoded in the formula labelling, by imposing the following type constraint:

573 (★) any two edges of \mathcal{N} which are supported by the same propositional variable lay in the
 574 same connected component of \mathcal{N}^{-b} .

575 If (★) holds (e.g. as for the proof-net \mathcal{N}_a discussed in Example 52), then $\ell_{\mathcal{N}}$ is equivalent to
 576 the renaming from the edge set-family to the variable set-family, an instance being given

577 by the renaming f mentioned in Example 33. Such a shortcut is however misleading in
 578 our setting, as the rules generating MLL proof-nets (Figure 1b) are more “granular” than
 579 the ones for Bayesian networks, in particular (\star) is not preserved by the $c(\mathcal{N}_{X^-}, X^-)$ rule
 580 (Example 41). A better alternative would be to introduce “term” variables, like the variables
 581 of simply typed λ -calculus, decorating edges.

582 6 The General Case

583 The results of the previous section can be extended to non-atomic proof-nets by using MLL
 584 cut-reduction. We just sketch here the main ideas, giving the details in the appendix. The
 585 reader can recall the proof-net \mathcal{N} in Figure 1e to follow the reasoning with an example.
 586 Given a MLL proof-net \mathcal{N} of conclusion Δ : (i) reduce \mathcal{N} to its normal form \mathcal{N}_0 by using
 587 the cut-reduction rules of Figure 1d. (ii) Decompose \mathcal{N}_0 into the syntax forest \mathcal{F}_Δ of its
 588 conclusions and the atomic sub proof-net \mathcal{N}_a of conclusions the atomic formulas $\text{At}(\Delta)$
 589 appearing in Δ , which are the leaves of \mathcal{F}_Δ . Notice that there is a bijection between $\llbracket \Delta \rrbracket$
 590 and $\llbracket \text{At}(\Delta) \rrbracket$, relating an element in $\vec{d} \in \llbracket \Delta \rrbracket$ with a tuple $\text{At}(\vec{d}) \in \llbracket \text{At}(\Delta) \rrbracket$ enumerating the
 591 atomic components of \vec{d} . (iii) Apply Algorithm 1 in order to compute $\ell_{\overline{\mathcal{N}}}(\llbracket \mathcal{N} \rrbracket)$. We have:

592 ► **Corollary 56.** *Let \mathcal{N} be a proof-net with conclusions Δ , and let \mathcal{N}_0 be the normal form of*
 593 *\mathcal{N} and $(\mathcal{F}_\Delta, \mathcal{N}_a)$ be the decomposition of \mathcal{N}_0 described above. For every $\vec{d} \in \llbracket \Delta \rrbracket$, we have:*

$$594 \quad \llbracket \mathcal{N} \rrbracket_{\vec{d}}^t = \ell_{\mathcal{N}}(\llbracket \mathcal{N}_a \rrbracket)_{\ell_{\mathcal{N}_a}(\text{At}(\vec{d}))}, \quad (5)$$

595 *if $\text{At}(\vec{d})$ agrees on $\ell_{\mathcal{N}_a}$, otherwise $\llbracket \mathcal{N} \rrbracket_{\vec{d}}^t = 0$.*

596 The cut-reduction in step (i) is linear in the size of \mathcal{N}_0 as MLL cut-reduction shrinks the size
 597 of a proof-net. Also the construction of \mathcal{N}_a out of \mathcal{N}_0 , and the read of $\ell_{\mathcal{N}_a}(\text{At}(\vec{d}))$ out of
 598 $\vec{d} \in \llbracket \Delta \rrbracket$ are linear. So all the complexity of this procedure is the calculation of $\ell_{\mathcal{N}}(\llbracket \mathcal{N}_a \rrbracket)$
 599 which has been discussed in the previous section.

600 7 Conclusion and Perspectives

601 We considered weighted relational semantics just as an instance of quantitative semantics, but
 602 these techniques can be applied verbatim to other web-based semantics, such as probabilistic
 603 coherence spaces [6] or finiteness or Köthe sequence spaces [11, 10].

604 One can wonder whether our results extend to richer linear logic fragments. The additive
 605 connectives \oplus and $\&$ can be reasonably added to the picture. In fact, by adopting some form
 606 of additive boxes [9], one can revisit the sum-product algorithm as a refactorization of a
 607 proof-net modulo commutative additive cuts. The exponential modalities (so encompassing
 608 full simply typed probabilistic λ -calculus) are more challenging as they require infinite sets
 609 at the semantical level. This will deserve future investigation.

610 Related to the above point is the correspondence alluded to in Remark 54. We will detail
 611 in a future work how to map any run ρ of the sum-product algorithm on an atomic proof-net
 612 \mathcal{N}_0 into a non-atomic proof-net \mathcal{N}_ρ , which rewrites into \mathcal{N}_0 and such that the intermediate
 613 factors appearing in ρ correspond to sub-proof-nets of \mathcal{N}_ρ .

614 As mentioned by Remark 53, the performance of many exact inference algorithms, such
 615 as sum-product, depends on the order of the components to eliminate and the problem of
 616 finding optimal orders is known to be NP-hard [4]. Many heuristics have been given based
 617 on the graph-theoretical structure of Bayesian nets. One can wonder whether the additional
 618 proof-theoretical structure (e.g. switching paths, empires [18]) can suggest new heuristics.

619 — References —

- 620 1 Nick Benton, Gavin Bierman, Valeria de Paiva, and Martin Hyland. Linear lambda-calculus
621 and categorical models revisited. In E. Börger, G. Jäger, H. Kleine Büning, S. Martini, and
622 M. Richter, editors, *Proceedings of the Sixth Workshop on Computer Science Logic*, pages
623 61–84. Springer Verlag, 1993. URL: citeseer.ist.psu.edu/benton92linear.html.
- 624 2 Antonio Bucciarelli and Thomas Ehrhard. On phase semantics and denotational semantics:
625 the exponentials. *Ann. Pure Appl. Logic*, 109(3):205–241, 2001.
- 626 3 Kenta Cho and Bart Jacobs. Disintegration and bayesian inversion via string diagrams. *Math.*
627 *Struct. Comput. Sci.*, 29(7):938–971, 2019. doi:10.1017/S0960129518000488.
- 628 4 Gregory F. Cooper. The computational complexity of probabilistic inference using bayesian
629 belief networks. *Artificial Intelligence*, 42(2):393–405, 1990. doi:[https://doi.org/10.1016/
630 0004-3702\(90\)90060-D](https://doi.org/10.1016/0004-3702(90)90060-D).
- 631 5 Raphaëlle Crubillé. Probabilistic stable functions on discrete cones are power series. In Anuj
632 Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on*
633 *Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 275–284. ACM,
634 2018. doi:10.1145/3209108.3209198.
- 635 6 Vincent Danos and Thomas Ehrhard. Probabilistic coherence spaces as a model of higher-order
636 probabilistic computation. *Information and Computation*, 209(6):966–991, 2011.
- 637 7 Adnan Darwiche. Bayesian networks. In Frank van Harmelen, Vladimir Lifschitz, and Bruce W.
638 Porter, editors, *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial*
639 *Intelligence*, pages 467–509. Elsevier, 2008. doi:10.1016/S1574-6526(07)03011-8.
- 640 8 Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge Univer-
641 sity Press, 2009. URL: [http://www.cambridge.org/uk/catalogue/catalogue.asp?isbn=
642 9780521884389](http://www.cambridge.org/uk/catalogue/catalogue.asp?isbn=9780521884389).
- 643 9 Lorenzo Tortora de Falco. The additive mutilboxes. *Ann. Pure Appl. Log.*, 120(1-3):65–102,
644 2003. doi:10.1016/S0168-0072(02)00042-8.
- 645 10 Thomas Ehrhard. On Köthe sequence spaces and linear logic. *Math. Struct. Comput. Sci.*,
646 12:579–623, 2002.
- 647 11 Thomas Ehrhard. Finiteness spaces. *Math. Struct. Comput. Sci.*, 15(4):615–646, 2005.
- 648 12 Thomas Ehrhard. Differentials and distances in probabilistic coherence spaces. In Herman
649 Geuvers, editor, *4th International Conference on Formal Structures for Computation and*
650 *Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany*, volume 131 of *LIPICs*, pages
651 17:1–17:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.
652 FSCD.2019.17.
- 653 13 Thomas Ehrhard. Cones as a model of intuitionistic linear logic. In Holger Hermanns,
654 Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE*
655 *Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages
656 370–383. ACM, 2020. doi:10.1145/3373718.3394758.
- 657 14 Thomas Ehrhard, Michele Pagani, and Christine Tasson. Probabilistic Coherence Spaces are
658 Fully Abstract for Probabilistic PCF. In P. Sewell, editor, *The 41th Annual ACM SIGPLAN-*
659 *SIGACT Symposium on Principles of Programming Languages, POPL14, San Diego, USA*.
660 ACM, 2014.
- 661 15 Thomas Ehrhard, Michele Pagani, and Christine Tasson. Full abstraction for probabilistic pcf.
662 *J. ACM*, 65(4), April 2018. doi:10.1145/3164540.
- 663 16 Thomas Ehrhard, Michele Pagani, and Christine Tasson. Measurable cones and stable, measur-
664 able functions: a model for probabilistic higher-order programming. *PACMPL*, 2(POPL):59:1–
665 59:28, 2018. URL: <http://doi.acm.org/10.1145/3158147>, doi:10.1145/3158147.
- 666 17 Thomas Ehrhard and Christine Tasson. Probabilistic call by push value. *Log. Methods Comput.*
667 *Sci.*, 15(1), 2019. doi:10.23638/LMCS-15(1:3)2019.
- 668 18 Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.
- 669 19 Jean-Yves Girard. A new constructive logic: classical logic. *Math. Struct. Comput. Sci.*,
670 1(3):255–296, 1991.

- 671 20 Jean-Yves Girard. Coherent banach spaces: a continuous denotational semantics. *Theor.*
672 *Comput. Sci.*, 227:297, 1999.
- 673 21 Jean-Yves Girard. Between logic and quantic: a tract. In Thomas Ehrhard, Jean-Yves Girard,
674 Paul Ruet, and Philip Scott, editors, *Linear Logic in Computer Science*, volume 316 of *London*
675 *Math. Soc. Lect. Notes Ser.* CUP, 2004.
- 676 22 Bart Jacobs, Aleks Kissinger, and Fabio Zanasi. Causal inference by string diagram surgery.
677 In Mikolaj Bojanczyk and Alex Simpson, editors, *Foundations of Software Science and*
678 *Computation Structures - 22nd International Conference, FOSSACS 2019, Held as Part of the*
679 *European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech*
680 *Republic, April 6-11, 2019, Proceedings*, volume 11425 of *Lecture Notes in Computer Science*,
681 pages 313–329. Springer, 2019. doi:10.1007/978-3-030-17127-8_18.
- 682 23 Yves Lafont. From proof nets to interaction nets. In Jean-Yves Girard, Yves Lafont, and
683 Laurent Regnier, editors, *Advances in Linear Logic*, volume 222 of *London Math. Soc. Lect.*
684 *Notes Ser.*, pages 225–247, 1995.
- 685 24 Jim Laird, Giulio Manzonetto, Guy McCusker, and Michele Pagani. Weighted relational
686 models of typed lambda-calculi. In *28th Annual ACM/IEEE Symposium on Logic in Computer*
687 *Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*. IEEE Computer Society, June
688 2013.
- 689 25 François Lamarche. Quantitative domains and infinitary algebras. *Theor. Comput. Sci.*,
690 94(1):37–62, 1992. doi:10.1016/0304-3975(92)90323-8.
- 691 26 Hugo Paquet. Bayesian strategies: probabilistic programs as generalised graphical models. In
692 Nobuko Yoshida, editor, *Programming Languages and Systems - 30th European Symposium on*
693 *Programming, ESOP 2021, Held as Part of the European Joint Conferences on Theory and*
694 *Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021,*
695 *Proceedings*, volume 12648 of *Lecture Notes in Computer Science*, pages 519–547. Springer,
696 2021. doi:10.1007/978-3-030-72019-3_19.
- 697 27 Judea Pearl. *Probabilistic reasoning in intelligent systems - networks of plausible inference*.
698 Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann, 1989.
- 699 28 Dario Stein and Sam Staton. Compositional semantics for probabilistic programs with exact
700 conditioning. *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*,
701 pages 1–13, 2021.
- 702 29 N. Zhang and D. Poole. A simple approach to bayesian network computations. In *Proceedings*
703 *of the 10th Biennial Canadian Artificial Intelligence Conference*, page 171–178. AAAI Press /
704 The MIT Press, 1994.

A

 Appendix

A.1 Auxiliary notions from Section 2 and 3

We recall the following notations:

► **Notation 57.** Given a formula F , we denote by $\text{At}(F)$ the sequent $X_1^\circ, \dots, X_n^\circ$ of the occurrences of the atomic sub-formulas appearing in F , enumerated from left to right. We extend this notation to sequents: $\text{At}(\Gamma) ::= \text{At}(\wp\Gamma)$. We recall that At defines a bijection between $\llbracket \Gamma \rrbracket$ and $\llbracket \text{At}(\Gamma) \rrbracket$. For example, if $\Gamma = X^+ \otimes (Y^- \wp Y^+), X^-$, then $\text{At}(\Gamma) = X^+, Y^-, Y^+, X^-$ and for any $((x_1, (y_1, y_2)), x_2) \in \llbracket \Gamma \rrbracket$, $\text{At}(((x_1, (y_1, y_2)), x_2)) = (x_1, y_1, y_2, x_2) \in \llbracket \text{At}(\Gamma) \rrbracket$.

We denote by \mathcal{F}_F the syntax tree of a formula F . Similarly, \mathcal{F}_Γ is the forest induced by the syntax trees of the occurrences of formulas in a sequent Γ . Notice that $\text{At}(\Gamma)$ gives an enumeration of the leaves of the trees in \mathcal{F}_Γ . Also, \mathcal{F}_Γ induces a proof-net of conclusions $\Gamma, \text{At}(\Gamma^\perp)$ by adding an axiom on the top of each leaf. We call this proof-net the axiom closure of \mathcal{F}_Γ and we denote it by $\overline{\mathcal{F}}_\Gamma$.

The following lemmata will be useful in the sequel.

► **Lemma 58** (canonical decomposition). A proof-net \mathcal{N} of conclusions Δ can be decomposed into the forest \mathcal{F}_Δ and a sub proof-net \mathcal{N}_a of conclusions $\text{At}(\Delta)$. We call the pair $(\mathcal{F}_\Delta, \mathcal{N}_a)$ the canonical decomposition of \mathcal{N} .

Proof (Sketch). By structural induction on the conclusions of \mathcal{N} , noticing that the axioms, boxes, weakenings and contractions are restricted to atomic formulas. ◀

► **Lemma 59.** Given a sequent Γ , recall the forest \mathcal{F}_Γ as defined in notation 57, so that its axiom closure $\overline{\mathcal{F}}_\Gamma$ is a proof-net of conclusions $\Gamma, \text{At}(\Gamma^\perp)$. For every $\vec{z} \in \llbracket \Gamma \rrbracket$, $\vec{z}' \in \llbracket \text{At}(\Gamma^\perp) \rrbracket$ we have:

$$\llbracket \overline{\mathcal{F}}_\Gamma \rrbracket_{(\vec{z}, \vec{z}')}^\iota = \begin{cases} 1 & \text{if } \text{At}(\vec{z}) = \vec{z}', \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

Proof (Sketch). By structural induction on Γ . ◀

► **Lemma 60** (Semantics on atoms). Let \mathcal{N} be an atomic proof-net with conclusions $\Gamma = X_1^\circ, \dots, X_n^\circ$ which has only one connected component and no semantical box. Then:

- the atomic formulas in Γ are all over the same propositional variable, i.e. $X_i = X_j$ for any $i, j \leq n$;
- for all $(x_1, \dots, x_n) \in \llbracket \Gamma \rrbracket$:

$$\llbracket \mathcal{N} \rrbracket_{(x_1, \dots, x_n)}^\iota = \begin{cases} 1 & \text{if } x_i = x_j \text{ for every } i, j \leq n, \\ 0 & \text{otherwise} \end{cases}$$

Proof. By structural induction on \mathcal{N} , remarking that the only possible nodes of \mathcal{N} are axioms, cuts, weakenings and contractions, as \mathcal{N} is atomic. The induction step is when there is a cut or a contraction splitting \mathcal{N} in two components $\mathcal{N}_1, \mathcal{N}_2$, then the induction hypothesis on the $\mathcal{N}_1, \mathcal{N}_2$ and the fact that the cut and contr cases of Figure 1c give non-zero coefficient only on equal indexes on the active edges, give the statement. ◀

A.2 Proofs of Section 3

Proof of Lemma 10. By induction on a sequentialisation of $\text{Cut}_\Gamma(\mathcal{N})$, using the cases of Figure 1c and distributing all the products over the sums generated by the cut bundle over Γ, Γ^\perp . ◀

744 **A.3 Proof of Section 4**

745 **Proof of Proposition 27.** The associativity, commutativity and neutrality of the factor
746 product are immediate from Definition 24.

747 As for property (1), by definition the two factors at the left and right sides of the equation
748 are over the same set-family $\mathbb{X} \cup \mathbb{Z}$. Let $\vec{x} \in \llbracket \mathbb{X} \cup \mathbb{Z} \rrbracket$, we have:

$$\begin{aligned}
749 \quad & \pi_{\mathbb{X} \cup \mathbb{Z}}(\pi_{\mathbb{X} \cup \mathbb{Y}}(\phi))_{\vec{x}} \\
750 \quad &= \sum_{\vec{y} \in \llbracket (\mathbb{X} \cup \mathbb{Y}) \setminus (\mathbb{X} \cup \mathbb{Z}) \rrbracket} \sum_{\vec{z} \in \llbracket \text{Fam}(\phi) \setminus (\mathbb{X} \cup \mathbb{Y}) \rrbracket} \phi(\vec{x}|_{\text{Fam}(\phi) \cap (\mathbb{X} \cup \mathbb{Y})}, \vec{y}|_{\text{Fam}(\phi)}, \vec{z}) \quad (a) \\
751 \quad &= \sum_{\vec{y} \in \llbracket \mathbb{Y} \setminus (\mathbb{X} \cup \mathbb{Z}) \rrbracket} \sum_{\vec{z} \in \llbracket \text{Fam}(\phi) \setminus (\mathbb{X} \cup \mathbb{Y}) \rrbracket} \phi(\vec{x}|_{\text{Fam}(\phi) \cap (\mathbb{X} \cup \mathbb{Y})}, \vec{y}|_{\text{Fam}(\phi)}, \vec{z}) \quad (b) \\
752 \quad &= \sum_{(\vec{y}, \vec{z}) \in \llbracket \text{Fam}(\phi) \setminus (\mathbb{X} \cup \mathbb{Z}) \rrbracket} \phi(\vec{x}|_{\text{Fam}(\phi) \cap (\mathbb{X} \cup \mathbb{Y})}, (\vec{y}, \vec{z})|_{\text{Fam}(\phi)}) \quad (c) \\
753 \quad &= \sum_{(\vec{y}, \vec{z}) \in \llbracket \text{Fam}(\phi) \setminus (\mathbb{X} \cup \mathbb{Z}) \rrbracket} \phi(\vec{x}|_{\text{Fam}(\phi)}, (\vec{y}, \vec{z})|_{\text{Fam}(\phi)}) \quad (d) \\
754 \quad &= \pi_{\mathbb{X} \cup \mathbb{Z}}(\phi)_{\vec{x}} \quad (e)
\end{aligned}$$

756 Line (a) unrolls the two projections following Definition 20. Line (b) simplifies the set where
757 the first summation ranges. Line (c) using the fact that $\mathbb{Y} \setminus (\mathbb{X} \cup \mathbb{Z})$ and $\text{Fam}(\phi) \setminus (\mathbb{X} \cup \mathbb{Y})$ give
758 a partition of $\text{Fam}(\phi) \setminus (\mathbb{X} \cup \mathbb{Z})$, by the hypothesis $\mathbb{Y} \subseteq \text{Fam}(\phi)$ and $\mathbb{Z} \cap \text{Fam}(\phi) = \emptyset$. Line
759 (d) is because $\vec{x}|_{\text{Fam}(\phi) \cap (\mathbb{X} \cup \mathbb{Y})} = \vec{x}|_{\text{Fam}(\phi)}$ since: $\text{Fam}(\phi) \cap (\mathbb{X} \cup \mathbb{Y}) = (\text{Fam}(\phi) \cap \mathbb{X}) \cup \mathbb{Y}$ and we
760 know by hypothesis that $\vec{x} \in \llbracket \mathbb{X} \cup \mathbb{Z} \rrbracket$. Finally, line(e) applied the definition of projection.

761 Concerning property (2), it is trivial to prove that the both factors are over the set-family
762 \mathbb{X} , because of the hypothesis $\text{Fam}(\psi) \subseteq \mathbb{X}$. So, let us take $\vec{z} \in \llbracket \mathbb{X} \rrbracket$, we have:

$$\begin{aligned}
763 \quad & (\pi_{\mathbb{X}}(\phi \odot \psi))_{\vec{z}} = \sum_{\vec{y} \in \llbracket (\text{Fam}(\phi) \cup \text{Fam}(\psi)) \setminus \mathbb{X} \rrbracket} (\phi \odot \psi)_{(\vec{z}|_{\text{Fam}(\phi) \cup \text{Fam}(\psi)}, \vec{y})} = \sum_{\vec{y} \in \llbracket \text{Fam}(\phi) \setminus \mathbb{X} \rrbracket} (\phi \odot \psi)_{(\vec{z}|_{\text{Fam}(\phi) \cup \text{Fam}(\psi)}, \vec{y})} \\
764 \quad &= \sum_{\vec{y} \in \llbracket \text{Fam}(\phi) \setminus \mathbb{X} \rrbracket} \phi(\vec{z}, \vec{y})|_{\text{Fam}(\phi)} \psi_{\vec{z}|_{\text{Fam}(\psi)}} = \left(\sum_{\vec{y} \in \llbracket \text{Fam}(\phi) \setminus \mathbb{X} \rrbracket} \phi(\vec{z}|_{\text{Fam}(\phi)}, \vec{y}) \right) \psi_{\vec{z}|_{\text{Fam}(\psi)}} = (\pi_{\mathbb{X}}(\phi) \odot \psi)_{\vec{z}}
\end{aligned}$$

767 ◀

768 **Proof of Proposition 32.** It is immediate that $f^\circ(\vec{y})$ agrees on f , for any $\vec{y} \in \llbracket \mathbb{Y} \rrbracket$. Vice
769 versa, if \vec{x} agrees on f , then take a $\vec{y} \in \llbracket \mathbb{Y} \rrbracket$ such that for $y_{f(a)} = x_a$. Such a \vec{y} of course
770 exists but it might be not totally defined by the previous equation if there are some b not in
771 the image set of f . This is not the case if f is a surjective map from $\mathcal{I}(\mathbb{X})$ to $\mathcal{I}(\mathbb{Y})$. ◀

772 The following lemmata state useful properties of renamings that can be easily argued
773 from the above definitions.

774 ► **Lemma 61.** Let f be a bijective renaming from \mathbb{X} to \mathbb{Y} , then:

- 775 1. for any $\mathbb{X}' \subseteq \mathbb{X}$ and factor ϕ over \mathbb{X} , $f(\pi_{\mathbb{X}'}(\phi)) = \pi_{f(\mathbb{X}')} (f(\phi))$;
- 776 2. for any factors ϕ, ψ over sub set-families of \mathbb{X} , $f(\phi \odot \psi) = f(\phi) \odot f(\psi)$;
- 777 3. for any factors ϕ, ψ over \mathbb{X} , $\phi = \psi$ iff $f(\phi) = f(\psi)$.

778 ► **Lemma 62.** Given renamings f from \mathbb{X} to \mathbb{Y} and g from \mathbb{Y} to \mathbb{Z} , then for every factor ϕ
779 over \mathbb{X} , $(g \circ f)(\phi) = g(f(\phi))$.

780 A.4 Proofs of Section 5

781 ► **Lemma 63.** *Let \mathcal{N} be an atomic proof-net of conclusions Δ . For every $\vec{d} \in \llbracket \Delta \rrbracket$, we have*
 782 *that:*

- 783 1. *if \vec{d} does not agree on $\ell_{\mathcal{N}}$, then $\llbracket \mathcal{N} \rrbracket_{\vec{d}} = 0$,*
- 784 2. *if moreover \mathcal{N} has no semantical box, then if \vec{d} agrees on $\ell_{\mathcal{N}}$, then $\llbracket \mathcal{N} \rrbracket_{\vec{d}} = 1$.*

785 **Proof.** Let \mathcal{N}' be the proof-net obtained by expanding each conclusion of the boxes in \mathcal{N}
 786 with an ax cut redex (Figure 1d). This means that \mathcal{N}' can be seen as a proof-net obtained
 787 by a bunch of cuts between the conclusions of the proof-net $\mathfrak{b}(\mathcal{N})$, assembling all boxes in \mathcal{N} ,
 788 and the axiom closure $\overline{\mathcal{N}^{-\mathfrak{b}}}$ of $\mathcal{N}^{-\mathfrak{b}}$. Notice that, $\ell_{\mathcal{N}'}$ can be seen as a conservative extension
 789 of $\ell_{\mathcal{N}}$, in particular $\vec{d} \in \llbracket \Delta \rrbracket$ agrees with $\ell_{\mathcal{N}'}$ iff it agrees on $\ell_{\mathcal{N}}$. Moreover, $\llbracket \mathcal{N}' \rrbracket$ cut reduces
 790 to $\llbracket \mathcal{N} \rrbracket$, so by invariance of the weighted interpretation, $\llbracket \mathcal{N}' \rrbracket = \llbracket \mathcal{N} \rrbracket$. These observations
 791 and Lemma 10 reduce the claim to:

$$792 \quad \llbracket \overline{\mathcal{N}^{-\mathfrak{b}}} \rrbracket_{\vec{d}} = \begin{cases} 1 & \text{if } \vec{d} \text{ agrees on } \ell_{\mathcal{N}}, \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

793 The proof of this latter equation is by induction on the number of connected components of
 794 $\overline{\mathcal{N}^{-\mathfrak{b}}}$. If $\overline{\mathcal{N}^{-\mathfrak{b}}}$ is connected, then all conclusions in Δ are associated with the same index by
 795 $\ell_{\mathcal{N}}$. So \vec{d} “agrees on $\ell_{\mathcal{N}}$ ” means that \vec{d} is a tuple of multiple occurrences of the same element
 796 in $\iota(X)$. Equation (7) is then immediate from Lemma 60.

797 Otherwise, let $\mathcal{N}_1, \mathcal{N}_2$ be two non-empty subnets of $\overline{\mathcal{N}^{-\mathfrak{b}}}$ partitioning its connected
 798 components. Notice that Δ splits into Δ_1 and Δ_2 , and any $\vec{d} \in \llbracket \Delta \rrbracket$ into $\vec{d}_1 \in \llbracket \Delta_1 \rrbracket$ and
 799 $\vec{d}_2 \in \llbracket \Delta_2 \rrbracket$. Notice also that \vec{d} agrees on $\ell_{\mathcal{N}}$ iff \vec{d}_1 agrees on $\ell_{\mathcal{N}_1}$ and \vec{d}_2 agrees on $\ell_{\mathcal{N}_2}$. We
 800 can then conclude by applying the induction hypothesis separately to \mathcal{N}_1 and to \mathcal{N}_2 . ◀

801 **Proof of Proposition 47.** If \vec{d} does not agree on $\ell_{\mathcal{N}}$, then we have $\llbracket \mathcal{N} \rrbracket_{\vec{d}}^{\iota} = 0$ by Lemma 63.
 802 Otherwise, we have $\vec{d} = \ell_{\mathcal{N}}^{\circ}(\ell_{\mathcal{N}}^{\bullet}(\vec{d}))$ and so $\llbracket \mathcal{N} \rrbracket_{\vec{d}}^{\iota} = \ell_{\iota}(\llbracket \mathcal{N} \rrbracket)_{\ell_{\mathcal{N}}^{\bullet}(\vec{d})}$ by Definition 45. ◀

803 Lemmata 10 and 63 yield the following technical result which is needed in the proof of
 804 Theorem 48.

805 ► **Lemma 64.** *By recalling the notations of Lemma 10, let $\text{Cut}_{\Gamma}(\mathcal{N})$ be a proof-net of*
 806 *conclusions Δ that can be decomposed into a proof-net \mathcal{N} of conclusions $\Delta, \Gamma, \Gamma^{\perp}$ and a*
 807 *bundle of cuts between the formulas in Γ and Γ^{\perp} . Then, for every $\vec{d} \in \llbracket \Delta \rrbracket$, we have:*

$$808 \quad \llbracket \text{Cut}_{\Gamma}(\mathcal{N}) \rrbracket_{\vec{d}}^{\iota} = \sum_{\substack{\vec{c} \in \llbracket \Gamma \rrbracket \\ \vec{c} \text{ agrees on } \ell_{\text{Cut}_{\Gamma}(\mathcal{N})}}} \llbracket \mathcal{N} \rrbracket_{(\vec{d}, \vec{c}, \vec{c})}^{\iota}$$

809 **Proof.** By convenience, let us assemble in Γ the positive atomic formulas labelling the
 810 premises of the cuts in the bundle underlined by the claim, so that Γ^{\perp} contains the labels of
 811 the negative premises of these cuts. Consider the proof-net \mathcal{N}' obtained from \mathcal{N} by adding a
 812 contraction below each conclusion in Γ^{\perp} and then taking the axiom closure, so that \mathcal{N}' has
 813 conclusions $\Delta, \Gamma, \Gamma^{\perp}, \Gamma^{\perp}$. By using the rules of Figure 1c, we have that:

$$814 \quad \llbracket \mathcal{N}' \rrbracket_{(\vec{d}, \vec{c}_1, \vec{c}_2, \vec{c}_3)} = \begin{cases} \llbracket \mathcal{N} \rrbracket_{(\vec{d}, \vec{c}_1, \vec{c}_2)} & \text{if } \vec{c}_2 = \vec{c}_3, \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

815 We denote by $\text{Cut}_{\Gamma}(\mathcal{N}')$ the proof-net obtained from \mathcal{N}' by adding a bunch of cuts between
 816 Γ and one of the two occurrences of Γ^{\perp} . Notice that $\text{Cut}_{\Gamma}(\mathcal{N}')$ is a proof-net, because

4:22 Sum-Product for MLL

817 $\text{Cut}_\Gamma(\mathcal{N})$ is supposed to be a proof-net and the contractions added to \mathcal{N}' do not introduce
 818 any new switching cycle. Also, by Equation (8) and Lemma 10 we have that $\llbracket \mathcal{N} \rrbracket_{(\vec{d}, \vec{c}, \vec{c})} =$
 819 $\llbracket \text{Cut}_\Gamma(\mathcal{N}') \rrbracket_{(\vec{d}, \vec{c})}$, so that by again Lemma 10:

$$820 \quad \llbracket \text{Cut}_\Gamma(\mathcal{N}) \rrbracket_{\vec{d}} = \sum_{\vec{c} \in \llbracket \Gamma \rrbracket} \llbracket \mathcal{N} \rrbracket_{(\vec{d}, \vec{c}, \vec{c})} = \sum_{\vec{c} \in \llbracket \Gamma \rrbracket} \llbracket \text{Cut}_\Gamma(\mathcal{N}') \rrbracket_{(\vec{d}, \vec{c})}$$

821 By Lemma 63 the terms in the last sum are null whenever \vec{c} does not agree on $\ell_{\text{Cut}_\Gamma(\mathcal{N}')}$,
 822 which is equivalent to say that \vec{c} does not agree on $\ell_{\text{Cut}_\Gamma(\mathcal{N})}$: the former renaming being
 823 a conservative extension of the latter. We then conclude by restricting the sum to the \vec{c} 's
 824 which agree on $\ell_{\text{Cut}_\Gamma(\mathcal{N})}$, so getting the statement. \blacktriangleleft

825 **Proof of Theorem 48.** By definition of factor projection and renaming, the factors of the
 826 two sides of Equation (4) are over the same set-family $\ell_{\mathcal{N}}(\Delta)$. Let us prove that they have
 827 the same associated function. Let $\vec{d} \in \llbracket \ell_{\mathcal{N}}(\Delta) \rrbracket$. We have that:

$$828 \quad \begin{aligned} & (\pi_{\ell_{\mathcal{N}}(\Delta)}(\ell_{\mathcal{N}}(\llbracket \mathcal{N}' \rrbracket) \odot \ell_{\mathcal{N}}(\llbracket \mathcal{N}'' \rrbracket)))_{\vec{d}} \\ 829 &= \sum_{\substack{\vec{c} \in \llbracket \ell_{\mathcal{N}}(\Gamma) \rrbracket \\ = \llbracket \ell_{\mathcal{N}}(\Gamma^\perp) \rrbracket}} \left((\ell_{\mathcal{N}}(\llbracket \mathcal{N}' \rrbracket)) \odot (\ell_{\mathcal{N}}(\llbracket \mathcal{N}'' \rrbracket)) \right)_{(\vec{c}, \vec{d})} & (a) \\ 830 &= \sum_{\substack{\vec{c} \in \llbracket \ell_{\mathcal{N}}(\Gamma) \rrbracket \\ = \llbracket \ell_{\mathcal{N}}(\Gamma^\perp) \rrbracket}} (\ell_{\mathcal{N}}(\llbracket \mathcal{N}' \rrbracket))_{(\vec{c}, \vec{d}|_{\ell_{\mathcal{N}}(\Delta')}}) } (\ell_{\mathcal{N}}(\llbracket \mathcal{N}'' \rrbracket))_{(\vec{c}, \vec{d}|_{\ell_{\mathcal{N}}(\Delta'')}}) } & (b) \\ 831 &= \sum_{\substack{\vec{c} \in \llbracket \ell_{\mathcal{N}}(\Gamma) \rrbracket \\ = \llbracket \ell_{\mathcal{N}}(\Gamma^\perp) \rrbracket}} \llbracket \mathcal{N}' \rrbracket_{(\ell^*_{\mathcal{N}}(\vec{c}), \ell^*_{\mathcal{N}}(\vec{d})|_{\Delta'})} \llbracket \mathcal{N}'' \rrbracket_{(\ell^*_{\mathcal{N}}(\vec{c}), \ell^*_{\mathcal{N}}(\vec{d})|_{\Delta''})} & (c) \\ 832 &= \sum_{\substack{\vec{c} \in \llbracket \Gamma \rrbracket = \llbracket \Gamma^\perp \rrbracket \\ \vec{c} \text{ agrees on } \ell_{\mathcal{N}}} } \llbracket \mathcal{N}' \rrbracket_{(\vec{c}, \ell^*_{\mathcal{N}}(\vec{d})|_{\Delta'})} \llbracket \mathcal{N}'' \rrbracket_{(\vec{c}, \ell^*_{\mathcal{N}}(\vec{d})|_{\Delta''})} & (d) \\ 833 &= \llbracket \mathcal{N} \rrbracket_{\ell^*_{\mathcal{N}}(\vec{d})} = \ell_{\mathcal{N}}(\llbracket \mathcal{N} \rrbracket)_{\vec{d}} & (e) \end{aligned}$$

835 Line (a) is the definition of factor projection (Definition 20) and the remark that $\ell_{\mathcal{N}}(\Gamma) =$
 836 $\ell_{\mathcal{N}}(\Gamma^\perp)$. Line (b) applies the definition of factor product (Definition 24), while Line (c) is
 837 by Definition 45. Line (d) uses the fact that $\ell^*_{\mathcal{N}}$ is a bijection from $\llbracket \ell_{\mathcal{N}}(\Gamma) \rrbracket$ and the subset
 838 of points in $\llbracket \Gamma \rrbracket$ which agree on $\ell_{\mathcal{N}}$ (Notation 43). Line (e) applies Lemma 64 and then
 839 Definition 45. \blacktriangleleft

840 **Proof of Theorem 49.** The proof is by induction on the number of boxes in \mathcal{N} . If there is
 841 no box, then the factor product over $\mathbf{b}(\mathcal{N})$ is the empty factor $(\emptyset, 1)$, and so its projection to
 842 $\ell_{\mathcal{N}}(\Delta)$ associates 1 with every element in $\llbracket \ell_{\mathcal{N}}(\Delta) \rrbracket$. We then conclude by Lemma 63 and
 843 Definition 45.

844 Otherwise, let \mathbf{b} be a semantical box of \mathcal{N} . Consider the proof-net \mathcal{N}^* obtained by
 845 expanding the conclusions Γ of \mathbf{b} with axiom cut redexes (Figure 1d), so to have \mathcal{N}^* of the
 846 form $\text{Cut}_\Gamma(\mathcal{N}', \mathcal{N}'')$ with \mathcal{N}' being \mathbf{b} and its conclusions Γ and \mathcal{N}'' the axiom closure of the
 847 sub-graph given by \mathcal{N} without \mathbf{b} , so that \mathcal{N}'' has conclusions Γ^\perp, Δ .

848 Notice that $\mathbf{Fam}_e^t(\mathcal{N}^*) \supseteq \mathbf{Fam}_e^t(\mathcal{N})$ and there is a renaming bijection f from $\mathbf{Fam}_e^t(\mathcal{N}^*)$
 849 to $\mathbf{Fam}_e^t(\mathcal{N})$ such that the composition $f \circ \ell_{\mathcal{N}^*}$ restricted to $\mathbf{Fam}_e^t(\mathcal{N})$ is equal to $\ell_{\mathcal{N}}$, as
 850 the axiom cut expansions generating \mathcal{N}^* do not equate any connected components of $\mathcal{N}^{-\mathbf{b}}$.
 851 Finally, since \mathcal{N}^* cut reduces to \mathcal{N} , the invariance of the weighted interpretation gives also
 852 $\llbracket \mathcal{N}^* \rrbracket = \llbracket \mathcal{N} \rrbracket$

853 We can therefore prove the equation in the statement for \mathcal{N}^* and then the equality follows
 854 for \mathcal{N} by applying f to the two sides of the equation and using Lemma 61 and Lemma 62.
 855 Moreover, in order to prove the claimed equation for \mathcal{N}^* , we need to remark that there is an
 856 obvious renaming bijection g from $\text{Fam}_c^t(\mathcal{N}'')$ to $\text{Fam}_c^t(\mathcal{N}^*)$, such that $g \circ \ell_{\mathcal{N}''}$ is equal to the
 857 restriction of $\ell_{\mathcal{N}^*}$ to $\text{Fam}_c^t(\mathcal{N}'')$. So:

$$\begin{aligned}
 859 \quad \ell_{\mathcal{N}^*}(\llbracket \mathcal{N}^* \rrbracket^t) &= [a] \pi_{\ell_{\mathcal{N}^*}(\Delta)}(\ell_{\mathcal{N}^*}(\iota(\mathbf{b})) \odot \ell_{\mathcal{N}^*}(\llbracket \mathcal{N}'' \rrbracket)) = [b] \pi_{\ell_{\mathcal{N}^*}(\Delta)}(\ell_{\mathcal{N}^*}(\iota(\mathbf{b})) \odot g(\ell_{\mathcal{N}''}(\llbracket \mathcal{N}'' \rrbracket))) \\
 860 \quad &= [c] \pi_{\ell_{\mathcal{N}^*}(\Delta)}(\ell_{\mathcal{N}^*}(\iota(\mathbf{b})) \odot g(\pi_{\ell_{\mathcal{N}''}(\Gamma^\perp, \Delta)}(\bigodot_{\mathbf{b}' \in \mathbf{b}(\mathcal{N}'')} \ell_{\mathcal{N}''}(\iota(\mathbf{b}'))))) \\
 861 \quad &= [d] \pi_{\ell_{\mathcal{N}^*}(\Delta)}(\ell_{\mathcal{N}^*}(\iota(\mathbf{b})) \odot \pi_{\ell_{\mathcal{N}^*}(\Gamma^\perp, \Delta)}(\bigodot_{\mathbf{b}' \in \mathbf{b}(\mathcal{N}'')} \ell_{\mathcal{N}^*}(\iota(\mathbf{b}'))))) \\
 862 \quad &= [e] \pi_{\ell_{\mathcal{N}^*}(\Delta)}(\pi_{\ell_{\mathcal{N}^*}(\Gamma^\perp, \Delta)}(\ell_{\mathcal{N}^*}(\iota(\mathbf{b})) \odot \bigodot_{\mathbf{b}' \in \mathbf{b}(\mathcal{N}'')} \ell_{\mathcal{N}^*}(\iota(\mathbf{b}'))))) = [f] \pi_{\ell_{\mathcal{N}^*}(\Delta)}(\bigodot_{\mathbf{b}' \in \mathbf{b}(\mathcal{N})} \ell_{\mathcal{N}^*}(\iota(\mathbf{b}'))))
 \end{aligned}$$

864 Equation (a) is by Theorem 48. Equation (b) is by the fact that $\ell_{\mathcal{N}^*} = g \circ \ell_{\mathcal{N}''}$ and by applying
 865 Lemma 62. Equation (c) then applies the induction hypothesis to \mathcal{N}'' and Equation (d) comes
 866 back to $\ell_{\mathcal{N}^*}$ by invoking Lemma 61 and Lemma 62. Equation (e) uses Proposition 27.(2)
 867 and the fact that $\text{Fam}(\ell_{\mathcal{N}^*}(\iota(\mathbf{b}))) = \ell_{\mathcal{N}^*}(\Gamma) = \ell_{\mathcal{N}^*}(\Gamma^\perp) \subseteq \ell_{\mathcal{N}^*}(\Gamma^\perp, \Delta)$. Finally Equation (f)
 868 uses Proposition 27.(1) and the fact that $\mathbf{b}(\mathcal{N}) = \mathbf{b}(\mathcal{N}^*) = \mathbf{b}(\mathcal{N}'') \uplus \{\mathbf{b}\}$. \blacktriangleleft

869 **Proof of Theorem 51.** We prove that the loop in line 5 preserves the invariants:

870 1. $\text{Fam}(\bigodot_{\phi \in F} \phi) = \text{Fam}_c^t(\mathcal{N}) \setminus \{C \in \omega \mid C \text{ processed}\}$,

871 2. $\pi_{\ell_{\mathcal{N}}(\Delta)}(\bigodot_{\phi \in F} \phi) = \ell_{\mathcal{N}}(\llbracket \mathcal{N} \rrbracket^t)$.

872 The claim then follows as at the end of the loop, item 1 gives $\text{Fam}(\bigodot_{\phi \in F} \phi) \subseteq \ell_{\mathcal{N}}(\Delta)$, and
 873 so $\pi_{\ell_{\mathcal{N}}(\Delta)}(\bigodot_{\phi \in F} \phi) = \bigodot_{\phi \in F} \phi$ and we can conclude by item 2.

874 The proof of item 1 is an immediate consequence of the projection in line 8 and the fact
 875 that no factor in $F \setminus F_c$ contains the processed component C as an index.

876 The proof of item 2 is by induction on the number i of the current iteration of the loop.
 877 The initial case is given by Theorem 49. Otherwise, let F^i denote the content of variable F
 878 at the i -th iteration of the loop, for $i > 0$, and F^0 being the initial value as given in line 4.
 879 We have then, by denoting with $\mathbb{X} = \bigcup_{\phi \in F_c} \text{Fam}(\phi)$,

$$880 \quad \pi_{\ell_{\mathcal{N}}(\Delta)}(\bigodot_{\phi \in F^{i+1}} \phi) = \pi_{\ell_{\mathcal{N}}(\Delta)}\left(\left(\bigodot_{\phi \in F^i \setminus F_c} \phi\right) \odot \left(\pi_{\ell_{\mathbb{X} \setminus \{C\}}}(\bigodot_{\phi \in F_c} \phi)\right)\right) \quad (\text{a})$$

$$881 \quad = \pi_{\ell_{\mathcal{N}}(\Delta)}(\pi_{\ell_{\mathbb{X} \setminus \{C\}}}(\left(\bigodot_{\phi \in F^i \setminus F_c} \phi\right) \odot \left(\bigodot_{\phi \in F_c} \phi\right))) \quad (\text{b})$$

$$882 \quad = \pi_{\ell_{\mathcal{N}}(\Delta)}(\bigodot_{\phi \in F^i} \phi) = \ell_{\mathcal{N}}(\llbracket \mathcal{N} \rrbracket^t) \quad (\text{c})$$

884 Line (a) is by definition of the loop in lines 5-9. Line (b) uses Proposition 27.(2), and line (c)
 885 Proposition 27.(1), with the last equation given by the induction hypothesis. \blacktriangleleft

886 A.5 Proofs of Section 6

887 **Proof of Corollary 56.** By the strong normalisation and confluence of cut-reduction, \mathcal{N}_0 is
 888 well-defined and by the semantic invariance, $\llbracket \mathcal{N} \rrbracket = \llbracket \mathcal{N}_0 \rrbracket$. Moreover, by Lemma 59 and
 889 Lemma 60, $\llbracket \mathcal{N}_0 \rrbracket_{\vec{d}}^t = \llbracket \mathcal{N}_a \rrbracket_{\text{At}(\vec{d})}^t$. Finally, remark that \mathcal{N}_a is an atomic proof-net, so we conclude
 890 by applying Proposition 47 to \mathcal{N}_a . \blacktriangleleft