Lambdi Calculi through the Lens of Linear Logic — Delia Kesner

Lecture 1 - *June 30, 2025*

Course Introduction

Why lambda calculus We use lambda calculus because it is a *foundational model* of computation and is used in several programming languages. It has a minimal syntax with maximal expressiveness, and is the basis for type systems, evaluation strategies and semantics. It is still a fertile field for research e.g. interaction with category theory, game semantics and more.

What is Linear Logic Linear logic is a **resource aware logic** (it cares about how resources are handled), there exist no weakening or contraction in it. To recover weakening and contraction there are two **exponential modalities**: why not *?* and of course/bang *!*. Formulas must be marked with exponentials to be erased/weakened or duplicated/contracted. And this will result in **Multiplicative Exponential Linear Logic** (MELL).

Why linear logic But why do we use linear logic? It refines intuitionistic and classical logic, controls duplication and erasure and naturally captures resource-sensitive computation. It also unlocks new insights of computations e.g. proof-nets (a graphical syntax), call-by-name vs call-by-value as logical phenomena and implicit complexity and cost models. It also provides a fruitful perspective for revisiting old ideas, like Offering a fine-grained look at evaluation and typing, and inspiring new calculi and type systems.

Goals of the Course The goals of the lecture series are to explore lambda calculi variants inspired by linear logic [1], like their syntax, semantics, operational properties, implementations and subsuming frameworks. And understand the structure of computation through linear logic types (intersection/quantitative types) and provide resource-aware tools for modern theoretical research (observational equivalence, inhabitation, genericity).

[1] Will not go into linear logic but how one applies it.

Course outline:

- Day 1: Linear Logic Proof-Nets
- Day 2: A Lambda-Calculus Inspired from Linear Logic Proof-Nets
- Day 3: Intersection/Quantitative Types
- Day 4: A Subsuming Framework Inspired from Linear Logic
- Day 5: Observational Equivalence By Means of Intersection Types

Contents for Day 1

1 Multiplicative Linear Logic (MLL)

1.1 MLL Formulas

Atomic Formulas Let p denote a positive atomic formula. We use $\underline{p}$ (underlined p) to represent its negation, i.e., a negative atomic formula. The negation operator $\perp$ is defined as an involutive operation.

$$(p)^\perp = \underline{p}, \quad (\underline{p})^\perp = p$$

Grammar of MLL Formulas In Multiplicative Linear Logic (MLL), the *tensor* connective ($\otimes$) means "both resources are given together and must be used together," while the *par* connective ($\parr$) means "either resource may be used, but the choice is determined externally."

$$A ::= p \mid \underline{p} \mid A \otimes B \mid A \parr B$$

Negation Rules Negation in MLL flips the role of resources: it turns offering into demanding, and transforms joint usage ($\otimes$) into independent alternatives ($\parr$), and vice versa, always preserving the idea that negating twice brings you back to the original resource.

$$\begin{aligned} (p)^\perp &= \underline{p}, & (\underline{p})^\perp &= p \\ (A \otimes B)^\perp &= A^\perp \parr B^\perp, & (A \parr B)^\perp &= A^\perp \otimes B^\perp \\ (A^\perp)^\perp &= A \end{aligned}$$

1.2 MLL (Two-sided) Sequent Presentation

Sequents are of the form $\Gamma \vdash \Delta$. The sequent rules of Multiplicative Linear Logic (MLL) manage how formulas are introduced and manipulated under strict resource discipline. The tensor rules ($\otimes$) express simultaneous availability of resources, while par rules ($\parr$) capture choices made externally. Cut and axiom handle duality and identity, and permutation rules allow rearrangement without altering logical content—yet unlike classical logic, structural rules like weakening and contraction are absent, reflecting that each resource must be used exactly once.

$$\frac{}{A \vdash A}\ (\text{ax}) \qquad \frac{\Gamma \vdash A, \Delta \qquad A, \Gamma' \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ (\text{cut})$$

$$\frac{\Gamma, A, B, \Gamma' \vdash \Delta}{\Gamma, B, A, \Gamma' \vdash \Delta}\ (\text{perm L}) \qquad \frac{\Gamma \vdash \Delta, A, B, \Delta'}{\Gamma \vdash \Delta, B, A, \Delta'}\ (\text{perm R})$$

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \parr B \vdash \Delta, \Delta'}\ (\parr \text{ L}) \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \parr B, \Delta}\ (\parr \text{ R})$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}\ (\otimes \text{ L}) \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash A \otimes B, \Delta, \Delta'}\ (\otimes \text{ R})$$

1.3 MLL (Unilateral) Sequent Presentation

With MLL we use a unilateral sequential presentation. We take anything from the left hand side, move it to the right hand side and negate it. In this way, we formulate the same thing but in a unilateral form, for example:

$$\frac{}{\vdash A^{\perp}, A}\ (\text{ax}) \qquad \frac{\vdash A, \Gamma \qquad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta}\ (\text{cut}) \qquad \frac{\vdash \Gamma, A, B, \Delta}{\vdash \Gamma, B, A, \Delta}\ (\text{perm})$$

$$\frac{\vdash A, B, \Gamma}{\vdash A \parr B, \Gamma}\ (\text{par}) \qquad \frac{\vdash A, \Gamma \qquad \vdash B, \Delta}{\vdash A \otimes B, \Gamma, \Delta}\ (\text{tensor})$$

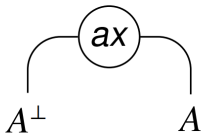
2 MLL Proof-Nets

An issue with this sequent presentation is that any possible sequent derivation captures a particulate constructor **history** (there are different possible sequentializations). Consider how many proofs ending in $\vdash (A_1 \parr A_2), ..., (A_{2n-1} \parr A_{2n})$ if we start from a derivation of $A_1, ..., A_{2n}$. We would have $n!$ possible derivations.

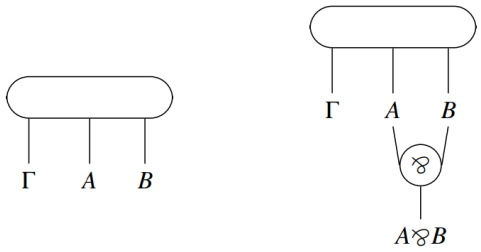
There is a better representation of proof derivation that abstracts such bureaucracy, namely, MLL Proof-Nets (PN).

An MLL PN with conclusions $A1, ..., A_n$ is a graph defined by induction as follows:

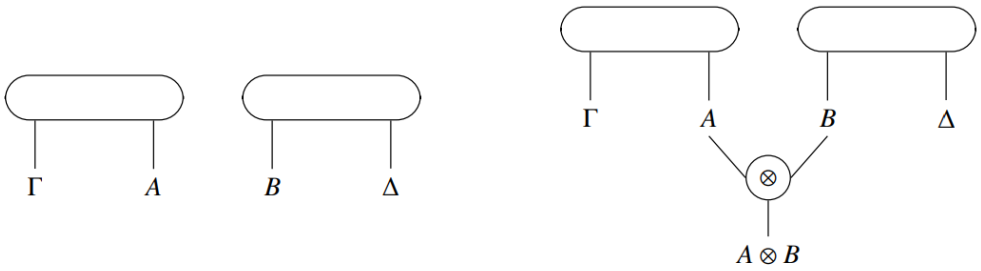
- For every MLL formula A we have a PN with conclusion $A^{\perp}, A$ having the form



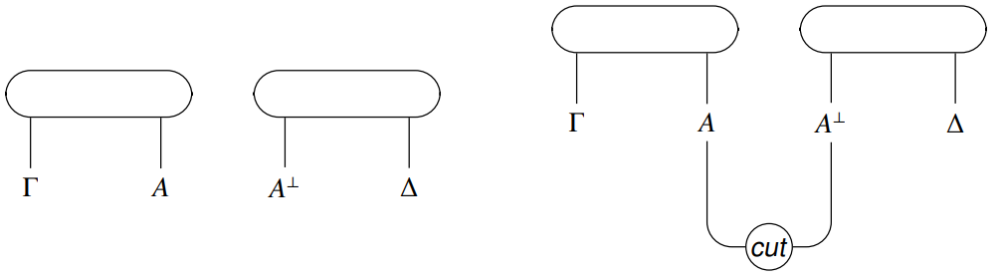
- Given a PN with conclusions Γ, A, B on the left we can construct the PN with conclusions $\Gamma, A \parr B$ on the right.



- Given a PN with conclusions Γ, A and a PN with conclusions Δ, B on the left, we can construct the PN with conclusions $\Gamma, A \otimes B, \Delta$ on the right.



- Given a PN with conclusions Γ, A and a PN with conclusions $\Delta, A^{\perp}$ on the left, we can construct the PN with conclusions Γ, Δ on the right.

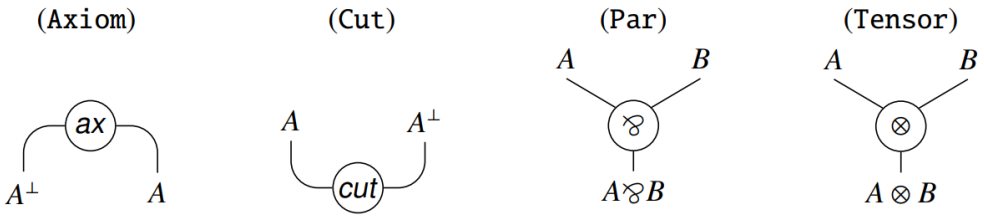


The nodes at the end of the graph is called the interface. Note that the order of appearance for the variables does not matter in the graphs i.e. $\Gamma, \Delta = \Delta, \Gamma$.

With proof nets we have no bureaucracy but also no history (no trace of the sequentializaiton/constructor history). In order to perform the cut proof above we need two individual sub proofs.

Pre-proof nets The pre proof nets (Pre-PN) solve this and are generated by the following links (see graph Fig below), and must satisfy the conditions:

- Every formula is the conclusion of exactly one link.
- Every formula is the premise of at most one link.



Note that a Pre-PN can be a valid PN.

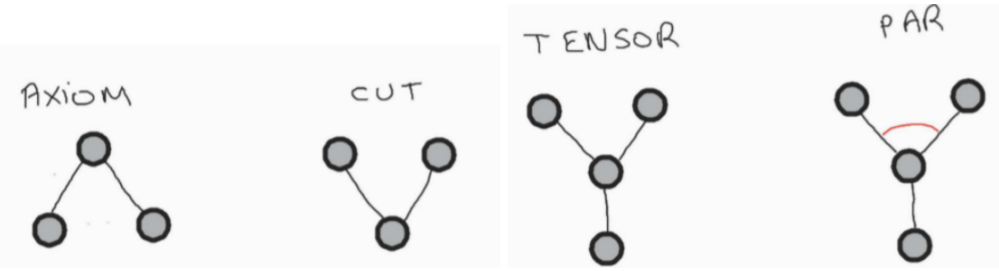
3 PN's Correctness criteria

How do we decide when something is or is not a proof net? With a **correctness criteria**, a method to determine if an arbitrary graph is a proof net or not. There are several methods for this: Long trip [1], acyclic-connected [2], and, contractibility [3].

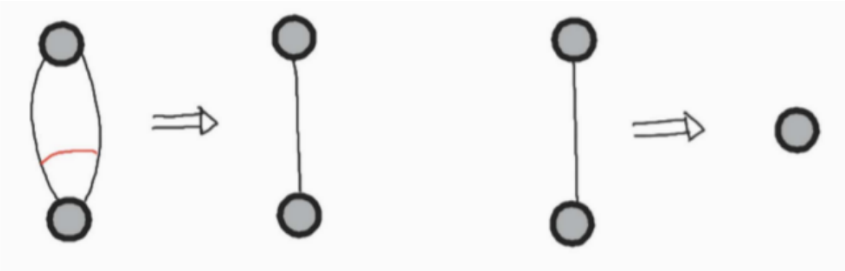
Long trip: A Pre-PN is a PN iff for every possible configuration of the links there is a unique long trip (i.e. a cycle visiting each node exactly once in each direction)

Acyclic-connected: A Pre-PN is abstracted by a paired graph S equipped with: a set V of vertices, a set E of edges and a set $C(S)$ of pairwise disjoint pairs of co-incident (one node in common) edges, marked with a red arc.

For every Pre-PN P we define a paired graph P^- by using the following constructions. We do not abstract away the operations and we only use the graph structure to determine correctness.



Contractability, we consider the following rewriting system with two rules.



Given a paired graph S:

- The first rule can only be applied to a pairwise disjoint pair of edges of $C(S)$ connecting the **same** two nodes
- The second rule can only be applied to an edge (not in $C(S)$) connection two **different** nodes

Theorem: a Pre-PN P is a PN iff P^{-} is contractible (i.e. reduces to a single node), it executes in quadratic time.

4 Multiplicative Exponential Linear Logic (MELL)

4.1 MELL Formulas

An expansion of MLL with the following formulas and sequent presentations, where we can weaken, contract, derelict, by explicitly adding an ! or ?

Atomic Formulas The atomic formulas for MELL is exactly the same as the ones in MLL. p is a positive atomic formula, whilst $\underline{p}$ is its negation, i.e. a negative atomic formula.

Grammar of MELL Formulas MELL extends MLL with two new connectives to include classical and intuitionistical logic's notion of using a formula any number of times. Hence, the unary connectives ! (named bang) and ? denote that a formula can be used for an unbounded number of times.

$$\underline{p} \mid A \mid A \otimes B \mid ?A \mid !A$$

Negation rules MELL include two additional negation rules that MLL does not have to include the new unary connectives (! and ?).

$$\begin{aligned} p^{\perp} &:= \underline{p} \\ (A \parr B)^{\perp} &:= A^{\perp} \otimes B^{\perp} \\ (?A)^{\perp} &:= !A^{\perp} \\ (A^{\perp})^{\perp} &:= A \end{aligned} \qquad \begin{aligned} \underline{p}^{\perp} &:= p \\ (A \otimes B)^{\perp} &:= A^{\perp} \parr B^{\perp} \\ (!A)^{\perp} &:= ?A^{\perp} \end{aligned}$$

4.2 MELL (Two-sided) inference rules

$$\frac{}{A \vdash A}\,\text{ax} \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,\text{cut}$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, B, A \vdash \Delta}\,\text{perm L} \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash B, A, \Delta}\,\text{perm R}$$

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \parr B \vdash \Delta, \Delta'}\,\text{par L} \qquad \frac{\Gamma \vdash A, \Gamma \qquad \Gamma \vdash B, \Delta}{\Gamma \vdash A \parr B, \Delta}\,\text{par R}$$

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \otimes B \vdash \Delta, \Delta'}\,\text{tensor L} \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \otimes B, \Delta}\,\text{tensor R}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, !A \vdash \Delta}\,!\ \text{weak} \qquad \frac{\Gamma, !A, !A \vdash \Delta}{\Gamma, !A \vdash \Delta}\,!\ \text{cont} \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta}\,!\ \text{der} \qquad \frac{!\Gamma, A \vdash ?\Delta}{!\Gamma, ?A \vdash ?\Delta}\,?\text{L}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash ?A, \Delta}\,?\ \text{weak} \qquad \frac{\Gamma \vdash ?A, ?A, \Delta}{\Gamma \vdash ?A, \Delta}\,?\ \text{cont} \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash ?A, \Delta}\,?\ \text{der} \qquad \frac{!\Gamma \vdash A, ?\Delta}{!\Gamma \vdash !A, ?\Delta}\,!\ \text{R}$$

Formulas starting with '!' operator may be copied or discarded on the left side of the $\vdash$. Formulas starting with '?' operator may be copied or discarded on the right side of the $\vdash$.

4.3 MELL (unilateral) inference rules

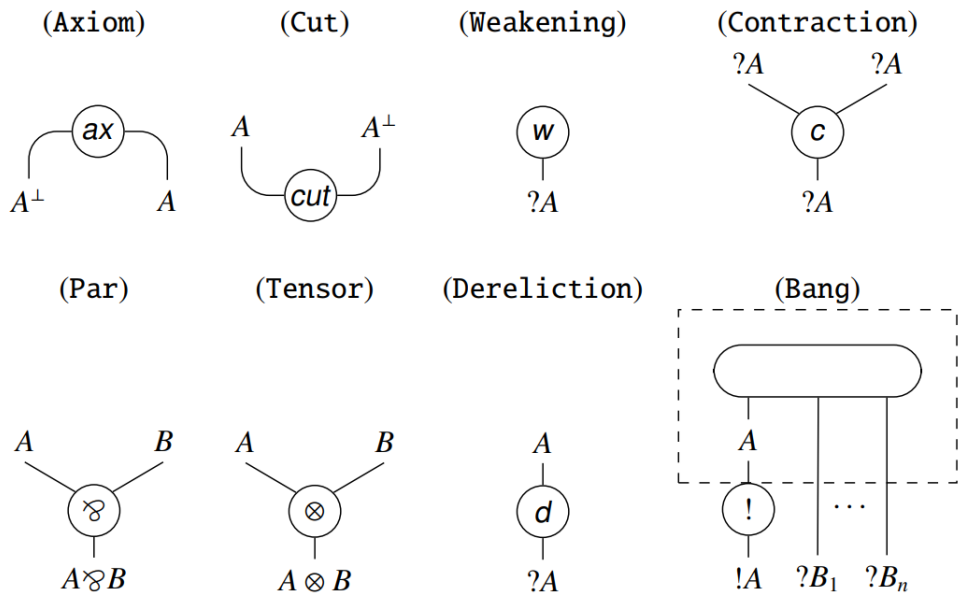
$$\frac{}{\vdash A^\perp, A}\,\text{ax} \qquad \frac{\vdash A, \Gamma \qquad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta}\,\text{cut} \qquad \frac{\vdash \Gamma, A, B, \Delta}{\vdash \Gamma, B, A, \Delta}\,\text{Perm} \qquad \frac{\vdash A, B, \Gamma}{\vdash A \parr B, \Gamma}\,\text{par}$$

$$\frac{\vdash A, \Gamma \qquad \vdash B, \Delta}{\vdash A \otimes B, \Gamma A, \Delta}\,\text{tensor} \qquad \frac{\vdash \Gamma}{\vdash ?A, \Gamma}\,\text{weak} \qquad \frac{\vdash ?A, ?A, \Gamma}{\vdash ?A, \Gamma}\,\text{cont}$$

$$\frac{\vdash A, \Gamma}{\vdash ?A, \Gamma}\,\text{cont} \qquad \frac{\vdash A, ?\Gamma}{\vdash !A, ?\Gamma}\,\text{cont}$$

4.4 MELL Proof-Nets

Here, we can also represent derivations in unilateral style as proof-nets too. To this end, we must expand the graphical syntax. For each MLL connective we just copy corresponding MLL rules. Now we just need to introduce one new rules for each of our new connective, that is '!' and '?'.



Dereliction allows us to always create a '?'.
Bang allows to put into a box (i.e. '!') every resource that does not depend on its context. Here we

need the rest of the context to be under '?', because A may later get copied and discarded, so the same thing would happen to the resources it depends on. Since we do not know which resources (which B_is) A depends on, we have to put all of them under the '?'.

A MELL PN with conclusions $A_1, ..., A_n$ is defined by induction as follows:

- Every MLL PN is a MELL PN
- Given a PN with conclusion Γ on the left, we can construct the PN with conclusions $\Gamma, ?A$
- Given a PN with conclusion $\Gamma, ?A, ?A$ on the left, we can construct the PN with conclusions $\Gamma, ?A$
- Given a PN with conclusions Γ, A on the left, we can construct the PN with conclusions $\Gamma, ?A$ on the right.
- Given a PN with conclusions $?\Gamma, A$, we can construct the following PN with conclusions $?\Gamma, !A$

4.5 Operational semantics

There are structural and cut elimination transformation. The structural transformations are equivalences and additional rewriting rules. The cut elimination have two sub-categories **multiplicative** rewriting rules (does not involve boxes) and **exponential** rewriting rules (involving boxes)

See slides p.44 – 53 at `https://www.cs.uoregon.edu/research/summerschool/summer25/_lectures/Kesner_Lesson1.pdf` for the color coded rewriting rules in sequent and graphical representations.

The reduction relations for MELL PN have the following relations

$$\mathcal{R} := C(a), C(), C(w, b), C(d, b), C(c, b), C(b, b), w - b, w - c$$
$$\mathcal{E} := A(c), IO(c)$$

where $C(x)$ are the cut elimination rules, $w - x$ are the rewriting rules and $\mathcal{E}$ holds all equivalence relations

Reduction relation is the closure by all PN context of the rules in $\mathcal{R}$. The congruence is the reflexive, symmetric, transitive, closed by PN contexts relations on PN generated by the equation $\mathcal{E}$. However, to guarantee terminations we need types.

4.6 MELL PN Properties

Definitions reduction relations:

- A reduction relation S is said to be confluent iff for every t, u, v such that $t \to_S^* u$ and $t \to_S^* v$ there us t' such that $u \to_S^* t'$ and $v \to_S^* t'$
- A reduction relation S is said to be **terminating** iff for every t there is no $\to_S$-sequence starting at t (i.e. every $\to_S$-reduction sequence starting at any term is terminating)
- A reduction relation S is said to be **strongly terminating** iff every **typed object** t is terminating

Theorem (confluence): The reduction $\to_{\mathcal{R}/\mathcal{E}}$ is confluent on MELL PN

Theorem (strong normalization): The reduction $\to_{\mathcal{R}/\mathcal{E}}$ is terminating on MELL PN (i.e. $\to_{\mathcal{R}/\mathcal{E}}$ is strongly normalizing)

References

[1] J.-Y. Girard, "Linear logic," *Theoretical Computer Science*, vol. 50, no. 1, pp. 1–101, 1987, ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(87)90045-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0304397587900454.

[2] V. Danos and L. Regnier, "The structure of multiplicatives," *Archive for Mathematical Logic*, vol. 28, no. 3, pp. 181–203, Oct. 1989, ISSN: 1432-0665. DOI: 10.1007/BF01622878. [Online]. Available: https://doi.org/10.1007/BF01622878.

[3] V. Danos, "La logique linéaire appliquée à l'étude de divers processus de normalisation (principalement du lambda-calcul)," 1990PA077188, Ph.D. dissertation, 1990. [Online]. Available: http://www.theses.fr/1990PA077188.