Università degli studi Roma Tre
Aix-Marseille Université

Master's Thesis in Mathematics

Correctness as Good Interactive Behavior

Adrien Ragot under the supervision of
Prof. Lorenzo Tortora de Falco.

Abstract

This Master's thesis is a reworking of Denis Béchet's article entitled "Minimality of the Correctness Criterion" [BEC98]. In 1998, Béchet questionned the meaning of the Danos-Regnier correctness criterion [DR89] for MLL proof nets relatively to cut elimination, successfully showing that it corresponds to a notion of good behavior (for structures without cuts). We offer a reworking of the article based on a more formal framework from graph theory, especially usefull in the – rather geometrical – argument of acyclity. But also for the treatment of cut–elimination we use preliminary concepts to ludics, that can be found in a paper of Curien [Cur05], hopefully obtaining more rigorous proofs. Given these formal tools we briefly question – as suggests Béchet in the original article – the possibility of an extension of the result to other fragments of linear logic.

Contents

The base of this thesis is an article written by Denis Béchet in 1998 and titled, "Minimality of the correctness criterion for multiplicative proof nets" see [BEC98]. We rework the article with more modern notions to describe the geometry of proof–nets but also the process of cut–elimination.

Proof–nets is a new representation of proofs that appeared with Girard linear logic, introduced in 1987 [Gir87]. In this representation proofs corresponds to graph[1] while the cut–elimination correspond to graph rewriting. The thing is that proof–nets belong to a more general class of graphs that are called proof–structures, and not all proof structures are proof–nets[2]. For this reason there has been a search for correctness criterion that characterize proof structures that are proof nets. The most famous for the MLL[3] fragment of linear logic is due to Danos and Regnier [DR89]. This criterion is a rather geometrical one based on a notion of switching[4], Béchet wondered in [BEC98] what this criterion would mean for cut–elimination. Béchet succesfully showed that DR[5]–correctness corresponds to a notion of good behavior relatively to cut–elimination for cut free proofs. The notion of *good behavior* is defined in the paper of Béchet, negatively, meaning he introduces a notion a *bad behavior* and then simply define good behavior as its negation. Proof–structures that behave badly are the one that interacting[6] with a correct proof–structure[7] reduce after cut elimination to a disconnected graph or a *deadlock*. A deadlock is an important proof structure that corresponds to an axiom node above a cut node.

Now we try to shed some light on what can motivate such a work. First why linear logic? Linear Logic can be seen as a refinement of classical logic, although discovered through denotational semantic of the λ–calculus, perhaps it's most important feature is the control of the rule of weakening and contraction of classical logic. The rules of weakening are indeed natural to consider in classical logic, the intuition in classical logic is that a sequent of the $\Gamma \vdash \Delta$ is intended to mean $\bigwedge \Gamma \to \bigvee \Delta$.

This intuition is no longer true in linear logic, where sequents are understood as $\bigwedge \Gamma \multimap \bigvee \Delta$[8]. The difference is born in what the linear implication $\multimap$ really is, namely, the transformation – or consumption – of a formula into another. [9]

[1] One can find the expression proofs–as–graphs.

[2] Meaning they represent a proof.

[3] This stands for Multiplicative Linear Logic.

[4] A transformation of the graph that may create disconnection, or leave cycles.

[5] This stands for Danos–Regnier.

[6] Here it means we link conclusions of the two graphs by a cut node.

[7] And so by characterization a proof–net.

[8] Here there is a slight abuse since in linear logic the connectors $\vee$and $\wedge$do not *really* occur, instead they are both decomposed in two connectors

[9] But still, the *classical implication* can be decomposed in linear logic as $A \to B = !A \multimap B$, using the linear implication $\multimap$ and the exponent !. To give an intuition, the exponentiation $!A$ can be understood as "there are infinitely many A's" or "A is an eternal truth".

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\, w-R \qquad \textit{and} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\, w-L$$

$$\frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\, c-R \qquad \textit{and} \qquad \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\, c-L$$

Linear logic is also following (one of) the intention of Gentzen sequent calculus for classical logic [Mél09]. What do we mean by this; the sequent calculus LK manipulates sequent of the form $\Gamma \vdash \Delta$ and the classical negation is generating a strong symmetry. This is expressed in the rule $(\neg_L - intro)$ and $(\neg_R - intro)$ of LK, these rules states that any formula A can pass on the other side of the sequent $\vdash$ by being negated, this also allows for one sided sequent in LK. This symmetry is just not present in intuistionistic logic.

$$\frac{\Gamma \vdash \Delta, A}{\Gamma, \neg A \vdash \Delta}\, \neg_L - intro \quad \textit{and} \quad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\, \neg_R - intro$$

Linear logic pursue this same direction, towards more symmetry. In fact through the control of weakening and contraction, it becomes apparent that there is a need to split the classic connectors $\wedge$ *"and"* and $\vee$ *"or"* in two versions, an additive and multiplicative one. Not doing so would not lead to a full control of the contraction and weakening there is a classic exemple with the use of the *"and"* connector[10]. In LL then, we have four connector $\otimes, \&$ the multiplicative and additive *"and"* and $⅋, \oplus$ the multiplicative and additive *"or"*. The symmetry again exhibited by the negation, it is that $\otimes$ and $⅋$ are duals, while $\&$ and $\oplus$ are duals. The idea is that negation does not change the multiplicative or additive character of a connector. This symmetry goes even further, also working with the units.

$$\frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\, \otimes \qquad \frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \,\&\, B}\, \&$$

What can motivate the proofs–as–graphs principle; with linear logic the new geometrical representation of proofs as a direct interest, it does not involve the unnecessary *bureaucraty* (an expression that comes from Girard) that can occur in the tree representation, for exemple the order on which we use a $⅋$ as conclusion. To reduce the bureaucracy is also a direction that one can go towards, since it means that we have less proofs to study : two proofs of the same sequent π_1, π_2 may have the same graphical representation, meaning that they only differ in bureaucratic aspects, not in logical ones. Indeed for proof theory this is an important feature to try to get a better understanding of what is *really* the logical content of a proof.

$$\frac{\dfrac{\vdash \Gamma, A, B, C}{\vdash \Gamma, A ⅋ B, C}\, ⅋}{\vdash \Gamma, (A ⅋ B) ⅋ C}\, ⅋ \quad \textit{is really similar to} \quad \frac{\dfrac{\vdash \Gamma, A, B, C}{\vdash \Gamma, A, B ⅋ C}\, ⅋}{\vdash \Gamma, A ⅋ (B ⅋ C)}\, ⅋$$

[10] When the context Γ and Δ are the same.

This representation also comes with a new object, the *proof structures*. They are a class of graphs of which some represents proofs of LL, which if the case occurs they are called proof nets. This apparent problem leads to a new question that is the search of correctness criterion to characterize proof structures that are proof nets. This new aspects draws stronger link between logic and graph theory, sure with Gentzen's work proofs where seen as trees, but the link with graph theory was still limited to this small class of graphs that are trees. The proof structures are a way more general class of graph, and in fact the correctness criterion has been shown to be related to a classic problem of graph theory that is the notion of *perfect matching*[11], see [Ngu20] and [Ret96].

Now we would like to draw the links between computer science and this work, this is – of course – based on cut elimination. One the strongest link between computer science and more specifically functional programming and proof theory is the Curry–Howard correspondence : the correspondence also known as the proofs–as–programs paradigm, states that proofs of Nm[12] corresponds to terms of the simply typed lambda calculus, while the formulas correspond to types and the cut elimination (or detour elimination for natural deduction) corresponds to the β–reduction, ie. the execution of programs.

$$\frac{\vdash \Gamma, A \quad \vdash \Delta, \neg A}{\vdash \Gamma, \Delta}\, cut$$

Since the discovery of this correspondence, logicians and computer scientist have tried to give a computational meaning to cut elimination with different deductive and typing systems. A successfull exemple is the correspondence that occurs with the system F and the natural deduction of the second order Intuistionistic logic, it has been shown by Girard and a formulation of this correspondence can be found in Proofs and Types [GTL89]. There was also attempt for the classical logic with Parigot's $\lambda\mu$–calculus in [Par92]. But note that it is still not always clear what cut elimination corresponds to computationally.

Among other things Linear Logic comes with a built–in notion of temporality that is not found in classical logic. This temporality is also corresponding to a conception of formulas as ressources, and the linear implication as a transformation of the ressource A into the ressource B. In fact there is a correspondance of differential proof nets and a form of enriched lambda calculus that is ressource sensitive, this types of calculus may be called ressource calculus. That kind of work can be found in [Tra11] or [Ehr16].

But with linear logic new ideas have emerged regarding the correspondence and the computational meaning of cut–elimination. Abramsky suggested that linear logic could be related to process calculus, and specifically the π–calculus, in what he presented the proofs–as–processes paradigm [Abr94], it has later been explore by others to name a few, Bellin and Scott in [BS94] and Beffara in [Bef14]. The great feature of this new correspondence, is that unlike for exemple

[11] A perfect matching of a graph corresponds to a set of edges without common vertices that matches each vertex of the graph, meaning each vertex is in the border of an edge.

[12] This denotes the so called Minimal logic, limited to the connector of implication $\rightarrow$ and eventually the connector "and" $\wedge$.

the $\lambda\mu$–calculus of Parigot, it extend the scopes of the correspondence, no more limiting it to functional programming, and so giving hope for a stronger link between proof theory and computer science (in this case to concurrency).

In the paper [BS94] cut elimination is correponding to a notion information flow and to sending/receiving protocols. Proof–structures are also involved in this new correspondence, and so is the theory of slicing that Girard introduced in [Gir87],[Gir96], it is related to MALL the multiplicative and additive fragment of linear logic. Such a correspondence would also be of interest in concurrency, since many processes calculus exists and such a correspondance could exhibit a candidate among the available calculus to keep, it would also bring the formal tools of proof theory to study problems of computer science.

The study of cut–elimination in LL proof–nets could therefore be a key ingredient towards a better understanding of the possible correspondence, but also eventually of results on concurrency or functional programming.

The work of Béchet is therefore in the line of what has been exposed with the Curry–Howard correspondence, involving the new notions that came with linear logic and the Geometry of interaction. Taking the new geometrical aspects of the correctness criterion for proof structures and questionning it relatively to cut elimination – that is the computational processes of proofs – this work aims towards getting a better understanding of the computational content of proofs, and doing so, of the new object that are the proof–structures.

The thesis is organized in the following way.

- The first three chapters are introductions to key notions for the understanding of the paper. We do not go as much as it would be needed into the details but try to record some important elements.
  - The section 1 is thought of as an incomplete introduction to proof theory, starting from the natural deduction of gentzen, we expose the notion of derivation as trees. We try to point out some differences between intuistionistic and classical logic, and expose the rules of inference of these deductive system (in natural deduction). We record the theorem of cut elimination for the sequent calculus of classical logic.
  - The section 2 is less related to the thesis but is here to illustrate the relation between the cut elimination and the β–reduction, ie. the executions of programs. We present the untyped λ–calculus and expose some classical results. We then present the simply typed lambda calculus as it can be seen in proofs and types (using the type generators $\rightarrow$ and $\times$). And we point out the relation between the natural deduction for intuistionistic logic and the simply typed lambda calculus, the well-known Curry–Howard correspondence.
  - The section 3 is a small presentation of linear logic, we point using traditionals exemple the differences between classical and linear logic, among others a 'built–in' notion of temporality. We then expose

the multiplicative fragment of linear logic denoted MLL, and the notion of MLL–proof–structures. We expose the need of correctness criterion, and present the one of Danos–Regnier (which is crucial to get into the article of D. Bechet [BEC98]). We also present the cut elimination, and point the structure that is the deadlock, which will also be of importance for us.

- The fourth section is still kind of introductory, it consists of notions of graph theory in order to describe more precisely the proof structures – and the modules – but, we aim to proove a theorem, which we choose to call the *Bechet's Cyclicity Criterion* a characterization of cycles in some types of graphs. We also present an upper bound in the number of successive meeting point of a family of arrows, which is a key element for a later proof of acyclity.
  - In the first subsection we expose elementary notions of graph theory, but more precisely of the theory of quivers, this object is used mainly in algebra. We define a notion of path in a quiver, and give a proposition to decompose path as composition – ie. correct enough concatenation – of descending and ascending path.
  - In this second subsection, we point out that a cycle can always be described in a standard way, meaning in a certain decomposition that we call standard.
  - In the third subsection, we define the notion of meeting points, in some specific graphs. Then we give an upper bound on the meeting points of a family of arrows with the right condition (compatible with modules) on the quiver. Finally we expose the cyclicity criterion in terms of meeting points. Then we give a formal definition of the modules using the exposed quiver theory, and call a definition presented in a paper of Nguyên [Ngu20], that is well compatible with the cyclicity criterion exposed.
- The fifth section is a formalisation of the notion of module (mainly for MLL), using the formalism of quivers presented in the previous part. We also define the notion of switching.
  - The first subsection is dedicated to giving the definition of modules, switching and proof structures using the previous quiver formalism. We also record the Danos-Regnier correctness criterion, althought we give no proof for it here, we invite the reader to have a look at [DR89] for a proof of the criterion.
  - The second subsection is dedicated to proving a theorem of completion. We seek to find correctness criterion for modules, in order to identify the modules that correspond to sub–modules of proof–nets. Such a criterion can be found, we choose to call it the DR^- correctness criterion and seek to show that it characterize the submodules

of proof–nets. This argument can be found in the original work of Béchet [BEC98], but we differ slightly on the presentation of the proofs, trying to make it more readable.

- The sixth section presents the notion of *bad behavior* as Bechet defined it in the original article [BEC98], then we present the $\parr$–closure and we show there that a proof behaves just like any of its $\parr$–closure, this is a commodity to study behaviors since then we can focus on only treating trees, instead of modules.

- The seventh section is a look at cut elimination for modules;

  – In its first subsection, we show that a definition of cut elimination using the notions of graph theory from the section 3, although realizable, ends up being too heavy.

  – In the second subsection, we present a (partial) solution that can be found in the paper of Curien [Cur05], the paper is an introduction to ludics, but first exposes some notions to get into ludics. In this framework module are seen as forest of formulaes representation, axioms corresponds to disjoint pairs of the occurency, while the cuts are disjoint pairs of relatives adresses. We can this way describe modules as expressions – but they may become heavy – and so cut elimination as rewriting.

- The eighth section treats the case of a proof structure with a disconnected switching σS, showing that with the right opponent the structure reduces to its switching σS.

- The ninth section treats the case of a proof structure with a switching cycle

  – In the first subsection, we expose the *bechet transformation*, that transform a module containing a minimal switching cycle into an acyclic module. But the transformation is not taking into account the types of the arrows, it is in the end more of geometrical argument and it is based on elements of the section 4.

  – In the second subsection we show that any acyclic configuration (ways to place the axiom) can be made into a well typed one. The argument is rather simple, it is based on the instantiation ϕ introduced by Bechet in the original article.

  – In the third subsection, we show that the confrontation with the created opponent will indeed reduce to a deadlock. To do so we use the notions of "pre-ludics" that we exposed in section 6.

- The tenth section briefly discuss of the possibility of a generalization to the MELL fragment, although it is rather incomplete, we want to point out a difficulty that could arise with contraction.

- The appendix A tries to generalize the lemme 9.1, that state that a minimal cyclic configuration can become an acyclic one. In fact we introduce a notion of minimal acyclicity and show that it is the dual notion of minimal cyclicity. We do not give an interesting meaning to minimal acyclicity, but exhibit some questions that can come with such a notion.

1 Gentzen's Natural deduction and Sequent Calculus

A starting point – Proofs representation. Going back to Gentzen – and probably much before him – there has been interest in trying to define what is a proof. In the end the question (for mathematicians) can arise pretty naturally, when can one say that he *proved* a proposition? When can two agree that a proposition has been proven? Gentzen introduced natural deduction in 1934 trying to give a formal definition of what is a proof. The fundamental idea is that proof are finite sequence of *'elementary'* deductions we could say, that are called *inference rules* or *logical rules*. Indeed there are classic exemples of these rules, one of the most natural one may be that from A and $A \to B$ one can conclude B, this rule is called the elimination rule of the implication and is denoted $(\to_e)$. This rule would then be represented as;

$$\frac{A \quad A \to B}{B} \to_e$$

Then the idea of Gentzen is to define a proof as a tree which has its nodes labeled by formulaes, and its edges labeled by rules. Although it did not stop here, the same year, Gentzen introduced the *sequent calculus* this was in order to proove consistency of Peano's Arithmetic PA – according to [Mél09] –. The main object introduced in sequent calculus are *sequents* they are ordered pairs (Γ, Δ) where Γ and Δ are multisets of formulaes. Sequents are denoted $\Gamma \vdash \Delta$, they are to be understand as $\bigwedge \Gamma \to \bigvee \Delta$, ie. the conjonction of the premisses ensures the disjunction of the conclusions. Proofs are then defined similarly to natural deduction, as trees which nodes labeled by sequents, and edges by rules.

$$\frac{\Gamma_1 \vdash \Delta_1, A \quad A, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} \text{ cut}$$

This last rule is similar to the modus ponens (adapted for sequents), it is called the *cut rule*. Gentzen based his proof of the consistency of Peano's arithmetic PA on a theorem of *cut-elimination* (Gentzen's Hauptsatz). Namely the theorem shows that to any proof (eventually using the cut rule) of some sequent $\Gamma \vdash \Delta$ corresponds a proof without cut of the same sequent.

1.1 First Order Logic

Before talking about proofs, we must first define the object that proofs manipulate. Proofs are ordering of propositions, intuitively a proposition is a sentence that is either true of false. We first define propositions. We will work either with propositional logic, or with first order logic. The formulas of propositional logic over a set of atomic formulas $\mathcal{V}$ is *generated* by the following grammar;

P, Q		Atomic formulas of a given set
$\bot, \top$		Logic constants
$A, B, C ::=$	$P, \top, \bot$	
	$\mid A \wedge B \mid$	Conjonction
	$\mid A \vee B \mid$	Disjonction
	$\mid A \to B \mid$	Implication

Figure 1: Formulaes of propositional logic.

The logic constants $\bot$ "bottom" and "top", they respectively correspond to "true" and "false".

Signature for first order logic. To define the proposition of first order logic, we first introduce the notion of signature Σ, it corresponds to a quadruplet $(\Sigma_F, \Sigma_R, \Sigma_C, ar)$ such that;

- Σ_R and Σ_F are called respectively the set of function symbols and of relation symbols.
- Σ_C is called the set of constant symbols.
- $ar : \Sigma_R \cup \Sigma_F \to \mathbb{N}$ is a function called *arity function*, which associates to each function and relation symbols an integer called, it's arity.

We will say that a function or a relation symbol s is of arity $n \in \mathbb{N}$ to mean that $ar(s) = n$. In the first order logic are, given a set of variables $\mathcal{V}$ and signature Σ. We first define the terms by the following grammar;

c		Constants symbols of a given set
$x, y, z, ...$		Variables
$t_1, ..., t_k ::=$	$c \in \Sigma_C, x \in \mathcal{V}$	
	$f(t_1, ..., t_k)$	Given a function symbol f of arity n.

Figure 2: Terms of first order logic.

P, Q		Atomic formulas of a given set
$\bot, \top$		Logic constants
$R(t_1, ..., t_k)$		Predicative judgements where $R \in \Sigma_R$ is of arity k
$A, B, C ::=$	$P, \top, \bot, R(t_1, ..., t_k)$	
	$\mid A \wedge B \mid$	Conjonction
	$\mid A \vee B \mid$	Disjonction
	$\mid A \to B \mid$	Implication
	$\mid \forall x.A \mid$	Universal quantification
	$\mid \exists x.A \mid$	Existential quantification

Figure 3: Proposition of first order logic.

1.2 Natural Deduction

Derivations in natural deduction. In natural deduction proofs might be called *derivation*, we will here define them. We define the *pre-derivations* as trees which arrows are labeled by formulaes of first order logic (eventually second). Given a pre-derivation $\mathcal{D}$, we will use the following notation to mean that $\mathcal{D}$ as a leaf (or more) labeled by A, and its root labeled by B.

$$\begin{array}{c} [A] \\ \vdots\ \mathcal{D} \\ B \end{array}$$

See in the next figure a graph representation of pre derivations, basically they are derivations without label consistency, ie the arrow can be labeled without restriction.

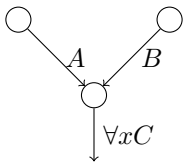


Figure 4: An exemple of a prederivation

We then will define a *derivation* or *deduction* of NJ a pre-derivation that is built by induction using the rules of NJ (see figure 7). The base of the induction are trees made of one node called *hypothesis node*, and one arrow labeled by any formula A, such a tree could be denoted $[A]$ or A. We also allows non leaf node to be linked to leaves by a *use arrow* (in the case of implication), we put a restriction on hypothesis nodes that they can only be the source of one use arrow. Note that we will use lighter notation for these trees, in the next figure we expose the two notations.

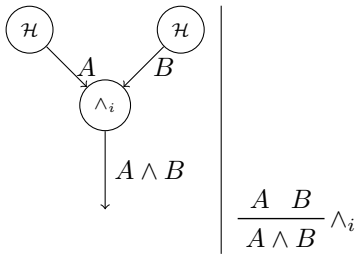


Figure 5: A natural deduction, in the graph representation, and the traditional representation

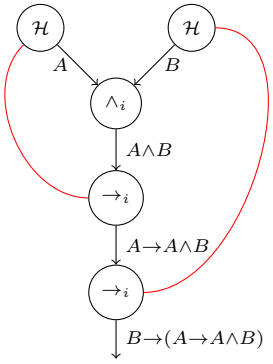
Active Hypothesis. Some rules can *use* leafs, meaning that the formulas are 'consumed' and no longer considered as hypothesis. Given a tree $\mathcal{D}$ labeled by rules and formulaes, denoting its set of leaves by $Leaf(\mathcal{D})$ that is equivalently the set of nodes labeled by $\mathcal{H}$. Then we define the *active hypothesis* the leaf nodes that are not linked to any node by a *reference link (arrow)*.

Formally $H_{\mathcal{D}} : \mathcal{F}(\mathcal{L}) \to \mathbb{N}$ is of domain $\mathcal{F}(\mathcal{L})$ formulaes of first order, and of codomain $\mathbb{N}$ the set of integers. Then the integer $H_{\mathcal{D}}(A)$ corresponds to the number of occurence of the label A in the conclusion of the active hypothesis (leaf) of $\mathcal{D}$. Formally it corresponds to $card(\{u \in Leaf(\mathcal{D}) \mid \ell(u) = A \wedge \mathrm{fl}(u)\})$.

We then might use the following notation to express that the root is labeled by A while the its active hypotheses correspond to the multiset Γ

$$\begin{array}{l} \Gamma \\ \vdots\ \mathcal{D} \\ \vdots \\ A \end{array}$$

Such A derivation $\mathcal{D}$ with its leaf labeled by the formulaes $B_1, ..., B_n$ and of root labeled by A, might be called a deduction of A under the hypothesis $B_1, ..., B_n$.



$$\dfrac{\dfrac{\dfrac{A \quad B}{A \wedge B}\wedge_i}{A \to (A \wedge B)}\to_i}{B \to (A \to (A \wedge B))}\to_i$$

Figure 6: A natural deduction involving references link

Formulaes as collections, Deductions as functions. With natural deduction, we want, given a formula A to try to provide an answer to the question *what is a proof of A?* Depending on the context, an old idea is to think of a formula A as the set of its possible deductions. Having this setup, one can say $\delta \in A$ for "δ prooves A".

In this traditional idea, we can then think of a deduction of A under the hypotheses $B_1, ..., B_n$ as a function $f : B_1, ..., B_n \to A$ that to any sequence of proofs $b_1, ..., b_n$ of respectively $B_1, ..., B_n$ associates a proof a of A.

Introduction Rules

$$\frac{A \quad B}{A \wedge B}\wedge_i \qquad \frac{A}{A \vee B}\vee_{i1} \qquad \frac{B}{A \vee B}\vee_{i2}$$

$$\frac{\begin{array}{c}[A]^1\\ \vdots\\ B\end{array}}{A \to B}\to_i(A,1) \qquad \frac{\begin{array}{c}[A]\\ \vdots\\ \bot\end{array}}{\neg A}\neg_i$$

$$\frac{A}{\forall x.A[x/a]}\forall_i \qquad \frac{A[t/x]}{\exists x.A}\exists_i$$

Elimination Rules

$$\frac{A \quad A \to B}{B}\to_e \qquad \frac{A \wedge B}{A}\wedge_{e1} \qquad \frac{A \wedge B}{B}\wedge_{e2}$$

$$\frac{A \vee B \quad \begin{array}{c}[A]^1\\ \vdots\\ C\end{array} \quad \begin{array}{c}[B]^2\\ \vdots\\ C\end{array}}{C}\vee_e \qquad \frac{A \quad \neg A}{\bot}\neg_e$$

$$\frac{\forall x.A}{A[t/x]}\forall_e \qquad \frac{\exists x.A \quad \begin{array}{c}[A[y/x]]^1\\ \vdots\ \mathcal{D}_0\\ C\end{array}}{C}\exists_e$$

Falsum Rules

$$\frac{\bot}{A}\bot_I \qquad \frac{\begin{array}{c}[A]^1\\ \vdots\\ \bot\end{array}}{\neg A}\bot_C$$

$$x, y \in \mathcal{V} \quad , \quad t \in \mathcal{T}$$

Figure 7: Natural deduction for intuistionistic propositional logic NJ (as given in Prawitz [Pra65]).

Restrictions. There is though some restrictions on the use of some of these rules (of figure 7),

- The introduction rules of the universal quantifier ($\forall_i$) may only be used if a does not occur in any assumption on which A depends.
- The elimination of the existantial quantifier ($\exists_e$) may only be used if the variable y does not occur in the formula $\exists x.A$ nor in any formula that labels a leaf of $\mathcal{D}_0$ other than $A[y/x]$.

Intuistionistic and classical logic. We want to stress the differences between intuistionistic and classical logic. The point is that intuistionistic logic is a constructive logic, this is mainly expressed throught the negation. In classical logic the negation is first syntaxing and formulas of the form $\neg A$ are formulaes of their own, while in intuistionistic logic $\neg A$ is merely a denotion for $A \to \bot$. In classical logic, the fact that A is not false (namely $\neg\neg A$) ensure the fact that A is true, this is rejected in intuistionistic logic. More precisely Intuistionistic logic reject the *law of excluded middle*, that states that $A \vee \neg A$ is always true.

In natural deduction, the difference of Intuistionistic logic and classical logic corresponds to the choice of permitted inference rules of the figure 7. In intuistionistic logic we consider all the rules of elimination and introduction, with the rule $\bot_I$, the system obtained is denoted NJ. On the other hand classical logic is made of all the rules of elimination and introduction and the rule $\bot_C$ – note that $\bot_I$ is derivable from $\bot_C$ –. The system obtained for classical logic is denoted NK. Therefore it appears that classical logic is stronger than intuistionistic logic, since a proof in NJ is a proof in NK (the reverse isn't true).

Towards sequent calculus. Given a derivation $\mathcal{D}$, recall that we denote by the following, the fact that the multiset of alive leaf-labels of $\mathcal{D}$ corresponds to Γ, while it's root is labeled by A. We introduce a new notation to express this property.

$$\begin{array}{l}\Gamma\\ \vdots\ \mathcal{D}\\ A\end{array} \qquad \text{or} \qquad \mathcal{D} : \Gamma \vdash A$$

This notation involves an ordered pair (Γ, A) where Γ is a multiset of formulas and A is a formula. This ordered pair is called a *sequent*. One can think of derivations as *construction* of a sequent, there could be several construction of the same sequent (understand deduction of A under the hypothesis Γ).

1.3 Detours

Minimal Logic. We will call minimal logic, the natural deduction limited to the set of rules $\{\wedge_i, \wedge_{e1}, \wedge_{e1}, \to_i, \to_e\}$, we might denote it **Nm**. The interest of the minimal logic can be found in its link with the simply typed lambda calculus, with the now famous correspondence of Curry Howard.

We can define an operation of graph rewriting in minimal logic, it will be denoted $\rightsquigarrow$, we call it *detour elimination*, Where a *detour* consists of a pattern made of the introduction of a rule immediatly followed by its elimination.

Definition 1.1 (Detours in natural deduction for minimal logic)**.** Given a natural deduction Π we call detour, any pattern that consists of introducing a rule and eliminating right away, so if Π contains a sub tree of the form $\dfrac{\dfrac{\begin{matrix}\vdots & \vdots \\ A & B\end{matrix}}{A \wedge B}\wedge_i}{A}\wedge_{e1}$, $\dfrac{\dfrac{\begin{matrix}\vdots & \vdots \\ A & B\end{matrix}}{A \wedge B}\wedge_i}{B}\wedge_{e2}$ or of the form $\dfrac{\begin{matrix}\vdots \\ A\end{matrix} \quad \dfrac{\begin{matrix}[A] \\ \vdots \\ B\end{matrix}}{A \to B}\to_i}{B}\to_e$ We say that Π contains a detour.

By the notation we mean that we consider the deduction under hypothesis , in which we replace every occurence of the hypothesis $[A]$ by an occurence of the deduction of A.

Definition 1.2 (Redexes in natural deduction)**.** Here are the redexes in natural reduction, they consist of suppressing a detour. That is why the relation $\rightsquigarrow$ could be called detour elimination. Given any derivation π_A of A and π_B of B we define the relation;

$$\dfrac{\dfrac{\begin{matrix}\vdots & \vdots \\ A & B\end{matrix}}{A \wedge B}\wedge_i}{A}\wedge_{e1} \rightsquigarrow \begin{matrix}\vdots \\ A\end{matrix} \qquad \dfrac{\dfrac{\begin{matrix}\vdots & \vdots \\ A & B\end{matrix}}{A \wedge B}\wedge_i}{B}\wedge_{e2} \rightsquigarrow \begin{matrix}\vdots \\ B\end{matrix}$$

$$\dfrac{\begin{matrix}\vdots \\ A\end{matrix} \quad \dfrac{\begin{matrix}[A] \\ \vdots \\ B\end{matrix}}{A \to B}\to_i}{B}\to_e \rightsquigarrow \begin{matrix}\vdots \\ A \\ \vdots \\ B\end{matrix}$$

Detour elimination in natural deduction limited to $\to, \wedge$.

Now note that $(\mathcal{ND}, \rightsquigarrow)$ is indeed a rewriting system since $\rightsquigarrow$ is indeed a binary relation over the set of natural deduction. From there using the definition we extend $\rightsquigarrow$ to a relation that commutes with the operating functions on $\mathcal{ND}$ that is to say the rule of introduction and elimination for the logic connectors. The commutating closure of $\rightsquigarrow$ will be the relation of detour elimination, it will be denoted $\rightsquigarrow$ if there is no ambiguity.

Definition 1.3 (Normal proof in natural deduction)**.** In natural deduction, a proof is said to be normal when it does not contain any sequence of introduction and elimination rule, ie. detours.

1.4 Sequent Calculus

As we mentionned before Gentzen introduced an object called *sequent*. If derivation of natural deduction correspond to the construction of *intuistionistic sequent* ie. with one conclusion, more generally a sequent is an ordered pair (Γ, Δ) often denoted $\Gamma \vdash \Delta$, where Γ and Δ are two multisets of formula. Intuitively they are to be understood as $\bigwedge \Gamma \Rightarrow \bigvee \Delta$, the conjonction of the premisses implies the disjunction of the conclusions.

If intuistionistic logic was more adapted to sequent calculus than was classical logic, with sequent calculus the case is the reverse. Classical logic enjoys a strong symmetry in sequent calculus (see [Mél09]) in fact in LK[13] the rules of negation are;

$$\frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\neg \mathrm{R}_{intro} \qquad \frac{\Gamma \vdash \Delta, A}{\Gamma, \neg A \vdash \Delta}\neg L_{intro}$$

It may be at first sight rather simple rules, but they have an important consequence (due to the right introduction); in LK sequents can be *one–sided* meaning they are of the form $\vdash \Gamma$, on the other hand this is simply not true for LJ (sequent calculus of intuistionistic logic) since intuistionistic sequent had one conclusion.

An important rule is the cut rule, it corresponds to the idea of *modus ponens* adapted for sequent calculus, in its one sided form it is the following.

$$\frac{\vdash \Gamma, A \quad \Delta, A}{\Gamma, \Delta}\, cut$$

Gentzen showed that in LK any proof π of a sequent $\vdash \Gamma$ can be transformed using a procedure called *cut elimination* into a proof π_0 that does not use the cut rule, the obtained proof is unique and is called the normal form of π. Gentzen originally showed it to proove the consistency of the arithmetic of peano, one can find a proof of the theorem (and more) in [HP96].

Theorem 1.1 (LK cut–elimination)**.** *Given a LK–proof π of a sequent S, π reduces into a proof π_0 of the same sequent S, denoted $\pi \rightsquigarrow \pi_0$, such that π_0 contains no cut.*

In fact it was even shown that the procedure of cut elimination is strongly normalizing in LK and LJ as it can be seen in [HP96] and [UB99]. The strong normalization indicates that no matter the strategy we apply the procedure of cut elimination will always terminate, ie. end up on a normalized proof.

[13] this denotes the sequent calculus for classical logic

2 The λ–calculus and the Curry–Howard correspondence

In this section, we introduce basic results on rewriting system and lambda calculus in order to briefly present the curry howard isomorphism. The interest of the Curry–Howard correspondence in our context is that, it gives a meaning of cut elimination in terms of computation, therefore giving a motivation to the work of Béchet.

2.1 Rewriting Systems

Definition 2.1 (Rewriting System)**.** We call rewriting system a couple $(\Sigma, \to)$ where;

- Σ is set of which elements are called *terms*.
- $\to$ is a binary relation on Σ called *rewriting*.

Given two terms t and u, we say that t reduces – or rewrites – in u if $t \to u$.

In the traditional bibliography rewriting systems are defined as a more general structure, here we limit it to the case of one rewriting relation, since we have no use for the general definition.

In general[14] the relation of rewriting correspond to the transitive closure of a binary relation $\to_0$. And a couple of terms (t, u) such that $t \to_0 u$ is called *redex*.

Confluent rewriting systems. We say that a rewriting system $(\Sigma, \to)$ is *confluent*, if given three terms u, v, z from $u \to v$ and $v \to z$ and $v \neq z$, it follows that there exists a term t such that $v \to t$ and $z \to t$. This property of the rewriting system is also called the Church–Rosser property.

On normalisation. A term t that contains that cannot be reduced[15] is said to be *normal* or *in normal form*. Also if a term t reduces to a term t_0 that is normal, we say that t_0 is *a normal form of* t, or that t *normalizes* in t_0. A term that has a normal form is said to be *normalizable*.

The notion of normal forms is key in rewriting systems and is not only occuring in the lambda calculus, in fact on can show that confluency ensures uniqueness of normal forms.

Proposition 2.1 (Confluency $\Rightarrow$ Uniqueness of normal forms)**.** *Assume we are given a confluent rewriting system* $(\Sigma, \to)$*, then any terms has at most one normal form.*

Proof. Assume that t has (at least) two normal forms, meaning that;

$$t \to t_0 \quad \text{and} \quad t \to t_1 \quad \text{and} \quad t_0 \neq t_1$$

[14] For exemple in λ–calculus.

[15] Meaning there exists no term u such that $t \to_\beta u$.

where t_0 and t_1 are normal, but the confluency ensures from this assumption that

$$\exists t_2 \in \Sigma(t_0 \to t_2 \quad \text{and} \quad t_1 \to t_2)$$

But this goes against t_0, t_1 being normal, therefore we have a contradiction. Therefore showing that a term t has at most one normal form. □

Definition 2.2 (Reduction chain)**.** Given t a term, we call *reduction chain* or *reduction strategy* of t, any sequence $\bar{t} : D \to A$ where $D \subset \mathbb{N}$, such that;

1. $D = \mathbb{N}$ or $D = \{0, ..., n\}$ for some integer n.
2. $t = t_0$ is the first term of the sequence.
3. Given an non null integer i in D, $t_{i-1} \to_\beta t_i$.

If the domain of the sequence D is finite we say that the chain is *finite*, otherwise we say that the chain is *infinite*. If the chain is finite we say it *finishes* in the term t_n.

If the chain is finite and its last term t_k is in normal form, we say that $\bar{t}$ is a *normalizing chain* or a *normalizing strategy*.

Strong and weak normalization. We distinguish to types of normalization, weak and strong normalization. A term is said to be

- *strongly normalizing* if any reduction strategy of t is normalizing.
- *weakly normalizing* if there exists one reduction strategy of t that is normalizing.

Note that being normalizable corresponds to being weakly normalizable. And that strong normalization ensures weak normalization.

2.2 The untyped lambda calculus

The Curry Howard correspondence. The Curry Howard correspondence has been discovered independantly in 1958 by Curry, and in 1969 by Howard, it is also known as the proofs-as-programs paradigm, it states the correspondence between proofs as defined in natural deduction by gentzen for intuistionistic logic **NJ**, and terms of the simply typed lambda calculus $\mathbf{\Lambda}_{\to,\times}$ – A rewriting system that is Turing complete (in the case of untyped lambda calculus) –. Furthermore the formulaes corresponds to types, and even more the rewriting of the simply typed lambda calculus, the β-reduction corresponds to the relation of cut-elimination that was introduced by Gentzen. Although we won't get exactly get into lambda calculus here, since Bechet's article merely focuses on proof structures, it is of interest to think of the curry howard correspondancy, since we will focus on giving a meaning relatively to cut elimination, ie. a dynamic between proofs, one can wonder what cut elimination (in proof structures) means computationally [PT10],[Bef14].

We will expose the untyped lambda calculus, expose some main results. Then we will expose the typing system that corresponds to the simply typed lambda calculus. Finally we will show how the correspondancy is defined.

The terms of the lambda calculus are defined, being given a set of variable symbol $\mathcal{V}$

$$t, u = x \in \mathcal{V} \quad | \quad (t)u \quad \textbf{application.} \quad | \quad \lambda x.t \quad \textbf{abstraction.}$$

where x is a variable symbol. A term of the form $(t)u$ is called an *application*, while a term of the form $\lambda x.t$ is called an *abstraction*. Note that this directly offers a principle of induction, since a term is either a variable, an application or an abstraction.

Free and bounded variables. Given a term t we can define its set of free variables and bounded variables by induction on the terms. the set of free variables of t is denoted $FV(t)$ and defined as;

$$FV(x) = \{x\} \quad | \quad FV(t.u) = FV(t) \cup FV(u) \quad | \quad FV(\lambda x.t) = FV(t) \setminus x$$

The intuition is that bounded variables are the one that are not free, but in practice it is not exact, as in fact a variable can be at the same time free and bounded in a term. The set of bounded variable is denoted $BV(t)$ and defined by induction;

$$BV(x) = \emptyset \quad | \quad BV(t.u) = BV(t) \cup BV(u) \quad | \quad BV(\lambda x.t) = BV(t) \cup x$$

The problem of a variable being simultaneously free and bounded in a term, can occur in an application $(t)u$, since it could be that a variable x is free in t but bounded in u. Consider the following exemple;

$$(x)\lambda x.y$$

Given a term t we might denote by $t[x_1, ..., x_n]$ to mean that $FV(t) \subset \{x_1, ..., x_n\}$.

Substitution and renaming. We define two kind of substitution one on free variables and one on bounded variables. We first define the substitution for free variables, we might call it simply *substitution*. Given $x_1, ..., x_n$ some variables and $t_1, ..., t_n$ some terms. For a term v we define the substitution of free variables as;

- $x[x_1/t_1, ..., x_n/t_n] = t_i$ if $x = x_i$ for some $1 \leq i \leq n$. Otherwise $x[x_1/t_1, ..., x_n/t_n] = x$.
- $(t)u[x_1/t_1, ..., x_n/t_n] = (t[x_1/t_1, ..., x_n/t_n])u[x_1/t_1, ..., x_n/t_n]$
- $\lambda x.t[x_1/t_1, ..., x_n/t_n] = \lambda x.t$ if $x = x_i$ for some $1 \leq i \leq n$. Otherwise $\lambda x.t[x_1/t_1, ..., x_n/t_n] = \lambda x.(t[x_1/t_1, ..., x_n/t_n])$.

From there we can define the substitution of bounded variables that we might call *renaming*, given $x_1, ..., x_n$ and $y_1, ..., y_n$ two collections of variables. We define the renaming by induction;

- $x\{x_1/y_1, ..., x_n/y_n\} = x$.
- $(t)u\{x_1/y_1, ..., x_n/y_n\} = (t\{x_1/y_1, ..., x_n/y_n\})u\{x_1/y_1, ..., x_n/y_n\}$
- $(\lambda x.t)\{x_1/y_1, ..., x_n/y_n\} = \lambda y_i.(t[y_i/x_i])$ if $x = x_i$ for some $1 \leq i \leq n$. Otherwise $(\lambda x.t)\{x_1/y_1, ..., x_n/y_n\} = \lambda x.(t\{x_1/y_1, ..., x_n/y_n\})$.

We might use the notation $t\{\overrightarrow{y}/\overrightarrow{x}\}$ and $t[\overrightarrow{y}/\overrightarrow{x}]$ for readability.

α–equivalency. On the terms of the untyped lambda calculus one can introduce the notion of α equivalency. We denote by $t \equiv_\alpha u$ the fact that two terms are α equivalent. This relation holds when by renaming the bounded variable of t we can obtain the term u.

Formally $t \equiv_\alpha$ if there exists families of variables $\overrightarrow{x_1}, ..., \overrightarrow{x_n}$ and $\overrightarrow{y_1}, ..., \overrightarrow{y_n}$ such that;

$$t\{\overrightarrow{y_1}/\overrightarrow{x_1}\}...\{\overrightarrow{y_n}/\overrightarrow{x_n}\} = u.$$

The rewriting, the β–reduction. The lambda calculus is first a rewriting system, it's notion of rewriting is called the β–reduction, it consists of only only one redex called a *head reduction*;

$$(\lambda x.t)u \succ t[u/x]$$

Then we allow for such a reduction to occur on any part of the term, namely the reduction is compatible with abstraction and applications, this way we can define the one step reduction that may be denoted $\rightarrow^0_\beta$;

- Given a term u, if $u \succ u'$ then $(t)u \rightarrow^0_\beta (t)u'$ for any term t.
- Given a term t, if $t \succ t'$ then $(t)u \rightarrow^0_\beta (t')u$ for any term u.
- Given a term t, if $t \succ t'$ then $\lambda x.t \rightarrow^0_\beta \lambda x.t'$ for any variable x.

Finally the β–reduction, denoted $\rightarrow_\beta$ is the transitive closure of $\rightarrow^0_\beta$. We might also use the notion of β–equivalency denoted $\equiv_\beta$ to refer to the transitive, reflexive and symmetric closure of $\rightarrow^0_\beta$.

Proposition 2.2 (The λ–calculus is Confluent)**.** *The lambda calculus in addition with its β–reduction $(\Lambda, \rightarrow_\beta)$ is a confluent rewriting system.*

The interest of the untyped lambda calculus is that it is a really expressive rewriting system and is in fact Turing complete, this means it can compute any computable function, or recursive function.

2.3 The simply typed lambda calculus

For the simply typed lambda calculus, we will follow the presentation given by Girard in proofs and types, therefore we add the following terms to the lambda terms.

$$\frac{u : A \quad v : B}{\langle u, v\rangle : A \times B}\times \qquad \frac{u : A \times B}{\pi_1(u) : A}\pi_1 \qquad \frac{u : A \times B}{\pi_2(u) : B}\pi_2$$

$$\frac{x : A \quad u : B}{\lambda x.u : A \to B}\text{abs} \qquad \frac{t : A \to B \quad u : A}{(t)u : B}\text{app}$$

Figure 8: Inference rules for the simply typed lambda calculus (as natural deduction).

$$\langle u, v\rangle \quad | \quad \pi_1(t) \quad | \quad \pi_2(t)$$

Along with the following redexes;

$$\pi_1\langle u, v\rangle \succ u \quad | \quad \pi_2\langle u, v\rangle \succ v$$

Types, and inference rules. Given a set of symbol $\{T_1, ..., T_n\}$ which elements are called *atomic types*. We define the types as;

$$\tau = T_i \mid \tau \times \tau' \mid \tau \to \tau'$$

The simply typed lambda calculus correspond to the presented lambda calculus to which we add a notion of *typing judgements*, they are expression of the form $t : A$ where t is a lambda term while A is a type. Then we introduce intuistionistic sequent $\Gamma \vdash t : A$ where Γ is a multiset of typing judgements that is called the *context*. Intuitively, such a sequent is valid, if from the fact that the typing judgement of Γ are true, the judgement $t : A$ is true.

Formally we introduce a set of rules alla Gentzen, to *derive* typing judgements.

There also exists a way to involve sequent calculus in typing systems. In this case we introduce intuistionistic sequent of for $\Gamma \vdash x : \sigma$ where Γ is a multiset of typing judgements. In such sequent the multiset Γ is called the *context*.

$$\frac{}{\Gamma, x : \sigma \vdash x : \sigma}\, ax \qquad \frac{\Gamma, x : \sigma \vdash u : \tau}{\Gamma \vdash \lambda x.u : \sigma \to \tau}\, abs$$

$$\frac{\Gamma \vdash \lambda t : \sigma \to \tau \quad \Gamma \vdash u : \sigma}{\Gamma \vdash (t)u : \tau}\, app$$

$$\frac{\Gamma \vdash u : \sigma \times \tau}{\Gamma \vdash \pi_1(u) : \sigma}\, \pi_1 \qquad \frac{\Gamma \vdash u : \sigma \times \tau}{\Gamma \vdash \pi_2(u) : \tau}\, \pi_2$$

$$\frac{\Gamma \vdash u : \sigma \quad \Gamma \vdash v : \tau}{\Gamma \vdash \langle u, v \rangle : \sigma \times \tau}\, and$$

Figure 9: Inference rules for the simply typed lambda calculus (as sequent calculus).

Typable terms. We then say that a term t of the lambda calculus is *typable* if and only if, there exists a type σ and a natural deduction of the typing judgement $t : \sigma$. Equivalently it means there exists a context Γ and a type σ and a type derivation of the sequent $\Gamma \vdash t : \sigma$. An important property is that typing is preserved by beta reduction (ie. execution).

Proposition 2.3 (Subject Reduction)**.** *Given two λ–terms u, v such that $u \to_\beta v$ ie. u reduces to v, we have the following implication;*

$$\Gamma \vdash u : \sigma \quad \Rightarrow \quad \Gamma \vdash v : \sigma$$

where Γ is a context, and σ is some type.

Proof. Have a look at [Kri02] or [SU10]. □

2.4 The correspondence of Curry–Howard

Range. Given a context $\Gamma = \{x_1 : \tau_1, ..., x_n : \tau_n\}$ we define the *range* of the context that is denote $\mid \Gamma \mid$ as the following multiset $\mid \Gamma \mid = \{\tau_1, ..., \tau_n\}$. Ie;

$$\mid \Gamma \mid = \{\tau \in \mathbb{T} \mid \exists x \in \mathcal{V}, x : \tau \in \Gamma\}.^{16}$$

Proposition 2.4 (Curry–Howard Correspondence in sequent calculus)**.** *Given Γ a context and σ a type we have the following equivalency;*

- *Given a term t, $\Gamma \vdash u : \sigma$ is derivable in $\Lambda_\to$ $\quad \Rightarrow \quad$ $\mid \Gamma \mid \vdash \sigma$ is derivable in Nm.*
- *$\mid \Gamma \mid \vdash \sigma$ is derivable in Nm $\quad \Rightarrow \quad$ There exists a term t such that $\Gamma_0 \vdash t : \sigma$ where $\Gamma_0 = \{(x_\tau : \tau) \mid \tau \in \Gamma\}$*

[16] There is a slight abuse in the ambiguity of the notions of set and of multiset.

Proof. Seen in [SU10]. □

CH with natural deduction. Perhaps the correspondence makes more sense with natural deduction, it is for exemple shown in Girard Proofs and types [GTL89]. Recall that essentially, a natural deduction under hypothesis Γ of a formula A is essentially a construction of the sequent $\Gamma \vdash A$. Using natural deduction, the correspondence operates between deductions and terms and by induction we can explicit a bijection, to do so we have to assume that the set of atomic proposition and atomic types are the same.

The typing derivation of a variable $x : A$ corresponds to a one node tree A of which its only node is labeled by A.

The typing derivation of an abstraction $\lambda x.u : \sigma \to \tau$ corresponds to the introduction of the implication.

$$\dfrac{\begin{array}{c}[x : A]\\ \vdots \\ u : B\end{array}}{\lambda x.t : A \to B}\, abs \qquad \textit{corresponds to} \qquad \dfrac{\begin{array}{c}[A]\\ \vdots \\ B\end{array}}{A \to B}\, \to_i$$

And finally the application corresponds to elimination of implication

$$\dfrac{t : A \to B \quad u : A}{(t)u : B}\, app \qquad \textit{corresponds to} \qquad \dfrac{A \to B \quad A}{B}\, \to_e$$

Then the key element is to notice that the elimination of the detour ($\to_i$ / $\to_e$) corresponds to the beta reduction;

$$\dfrac{\dfrac{\begin{array}{c}[A]\\ \vdots\ \mathcal{D}_1 \\ B\end{array}}{A \to B}\, \to_i \quad \begin{array}{c}\vdots\ \mathcal{D}_2 \\ A\end{array}}{B}\, \to_e \qquad \textit{reduces to} \qquad \begin{array}{c}\vdots\ \mathcal{D}_1 \\ A \\ \vdots\ \mathcal{D}_2 \\ B\end{array}$$

It becomes in the lambda calculus the following reduction;

$$\dfrac{\dfrac{\begin{array}{c}[x : A]\\ \vdots\ \mathcal{D}_1 \\ y : B\end{array}}{\lambda x.y : A \to B}\, abs \quad \begin{array}{c}\vdots\ \mathcal{D}_2 \\ z : A\end{array}}{(\lambda x.y)z : B}\, app \qquad \textit{reduces to} \qquad \begin{array}{c}\vdots\ \mathcal{D}_1 \\ z : A \\ \vdots\ \mathcal{D}_2 \\ y[z/x] : B\end{array}$$

Indeed it corresponds to the beta reduction, if one looks at the roots of two derivation trees. In fact with church typing the equivalency between proofs and λ–term alla Church is even clearer, since terms are directly typed, a term corresponds to a derivation of minimal logic Mn. One can look at Girard's work in proofs and types [GTL89] or the lecture of Sorensen and Urzyczyn [SU10].

3 Linear Logic, Geomatry of interaction

3.1 An Overview of Linear Logic

From Classical to Linear Logic. Most mathematicians are familiar with classical logic (denoted LK in its sequent calculus form), it deals with *eternal truths*, proposition (and their truths) are static, in the sense that we can infinitely use a proposition in a mathematical proof, this is expressed in the weakening and contraction rules of LK, they are said to be *structural rules*. In one sided sequent they are of the form;

$$\frac{\vdash \Gamma}{\vdash \Gamma, A}\,\mathrm{w} \qquad \frac{\vdash \Gamma, A, A}{\vdash \Gamma, A}\,\mathrm{c}$$

The weakening rule states – in particular – that if a proposition A has been proven then $A \vee B$ has been proven, indeed from the classical point of view there would be no problem in such a deduction. The staticity of classical logic goes even further though, as it can also be seen in multiplicatives and additives identity.

$$\frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Theta, B}{\Gamma \vdash \Delta, \Theta, A \wedge B}\,\wedge_a \qquad \frac{\Gamma \vdash \Delta, A \quad \Psi \vdash \Theta, B}{\Gamma, \Psi \vdash \Delta, \Theta, A \wedge B}\,\wedge_m$$

The first rule is said to be additive, which means the context of the sequent premisses are the same, the second rule is said to be multiplicative, since the context of the premisses are diverse. For exemple in classical logic the following derivation,

$$\frac{A \vdash B \quad A \vdash C}{A \vdash B \wedge C}\,\wedge_a$$

is a valid one in classical logic, and as it said in Okada's paper [Oka98], it is obviously true for mathematical reasoning. A problem – or merely a disconfort – arise with classical logic when we try to reason about ressources, there is a traditional exemple;

$$\frac{\text{U has 1 dollar} \vdash \text{U gets a cookie} \quad \text{U has 1 dollar} \vdash \text{U gets a coffee}}{\text{U has 1 dollar} \vdash \text{U gets a cookie} \wedge_a \text{U gets a coffee}}\,\wedge_a$$

This reasoning would be valid in classical logic. Although this does not mean that classical logic is wrong, but it stresses the fact that classical logic has no implicit notion of temporality, of dynamic, of states. If the statement "U has 1 dollar" is true then so are the two statements "U gets a cookie" and "U gets a coffee", in fact these statement could be then understood as "U can get a cookie" or "U can get a coffee", classical logic does not indicate in what state we are (If U got a cookie, if he got a coffee, or even if U spended his dollar), it merely indicates what is eternaly true. Although as it can be found in [Mél09], classical

logic *could* be able to express temporality with what is called temporal logic, but this approach ends being heavy and overall uncomfortable since formulaes find themselves indexed by a temporal index, for exemple $A_{t_0} \to B_{t_1}$.

Linear Logic. Linear Logic (**LL**) was introduced by Girard in 1987 in [Gir87] (discovered through denotational semantic). One of its key feature is that it is a *substructural logic*, by this we mean that the rules of contraction and weakening as we know them in classical logic, are not allowed. But more, linear logic also now distinguishes the additive and multiplicative connectors. One of the consequence is that now linear logic has a "built-in" notion of temporality, or one could say of current state. But if Linear logic seek to limit the weakening or contractions it does not totally reject them, as Girard puts it in his article [Gir95];

> *In linear logic, both contraction and weakening will be forbidden as structural rules. But linear logic is not logic without weakening and contraction [...] The main difference is that we now control in many cases the use of contraction, which – one should not forget it – means controlling the length of Herbrand disjunctions, of proof-search, normalization procedures, etc...*

Two exponents are introduced to control weakening and contraction. the "bang" or "of course" exponent "!", one can think of $!A$ has meaning that A is and eternal truth, ie. it can be used infinitely as a premisse – meaning the left sided weakening and contractions are allowed on formulaes $!A$ –. The other component "why not" denoted "?" is its symmetric counterpart, on a formula $?A$ we allow right sided weakening and contraction. To give an intuition of its meaning, one can think of $?A$ as $non(!non(A))$ which would denoted $!(A^{\perp})^{\perp}$, $?A$ can so be thought of the fact that A could be true at a certain state (and maybe 'never' or 'eternally'), since if $!(A^{\perp})$ holds this would ensure that A can be true at no state.

Another important point to notice, is how negation behaves in Linear logic, as opposed to intuistionistic logic, the negation allows for one sided sequent (that is also true in classical logic).

In Linear Logic the traditional ressource exemple becomes

$$\frac{\text{U has 1 dollar} \vdash \text{U gets a cookie} \quad \text{U has 1 dollar} \vdash \text{U gets a coffee}}{\text{U has 1 dollar} \vdash \text{U gets a cookie} \,\&\, \text{U gets a coffee}}\,\&$$

Where & is the additive connector, & as then to be understood as a choice, either the user U gets a coffee or a cookie.

3.2 Geometry of Interaction, Multiplicative linear logic

Multiplicative Linear Logic. In this document, we will not present the whole fragment of linear logic which can be heavy. We limit ourselves to the multi-

plicative fragment of linear logic – denoted **MLL** – meaning we only consider the *multiplicative* connectors $\parr$ (read 'parr') and $\otimes$ (read 'tensor'). We give these connectors the appelation multiplicative to mean that they are context independant. They behave respectively like an or and an and. We give in the next figure, a presentation of the formulas of **MLL**, given $\mathcal{V}$ a set of atomic formulas. The De Morgan laws of the *linear negation* – denoted $(.)^\perp$ – along with the rules that compose the fragment.

$$A, B = X \in \mathcal{V} \quad | \quad A^\perp \quad | \quad A \parr B \quad | \quad A \otimes B$$

$$(A \parr B)^\perp = A^\perp \otimes B^\perp \qquad (A \otimes B)^\perp = A^\perp \parr B^\perp$$

$$\frac{}{\vdash A, A^\perp}\,\mathrm{Ax} \qquad \frac{\vdash \Gamma, A \quad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\,\mathrm{Cut} \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\,\parr \qquad \frac{\vdash \Gamma, A \quad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta}\,\otimes$$

Figure 10: Formulas, De Morgan laws, and rules of the **MLL** fragment

Proof Structures. One can notice that this fragment is really small, which is a great asset to study it. A proof in **MLL** is as usually, a tree which nodes are labeled by formula's and edges are labeled by rules. We might call a proof of **MLL** a *sequential proof*. But here we will not study the usual proof representation of this logical system. **MLL** enjoys the good property of having proofs that are representable as graphs.

To represent proofs-as-graphs, we first introduce a class of graphs that are called *proof structure*. Proof structures are graphs (that may well be disconnected), and that are made of the 4 following links, see 11.

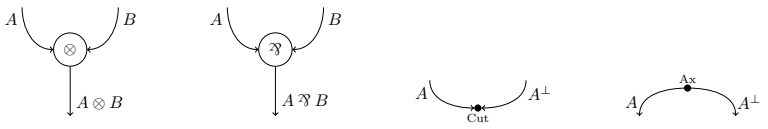


Figure 11: Links for the multiplicative proof structures.

Proof Nets. From there, define *proof nets* as proof structure that represent a proof. Formally we introduce a function [.] that sends each proof of **MLL** to a proof structure. This function is defined by induction, as in 12.

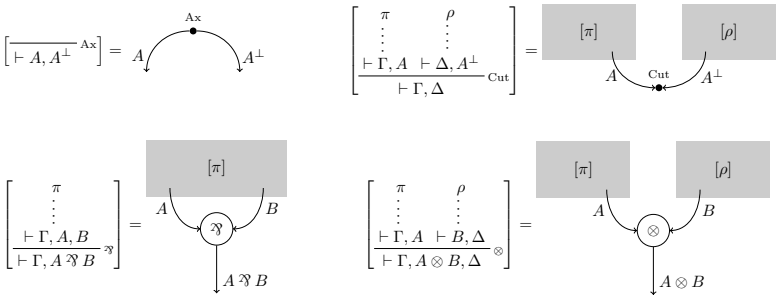


Figure 12: Representation of **MLL** proofs as proof structures

Definition 3.1 (Proof net)**.** A proof structure S is said to be a *proof net*, if and only if, S represents a proof. Ie. there exists a proof π such that $[\pi] = S$.

Correctness Criterion. But all proof structure are not proof nets; one of the obvious case would be the case of a disconnected proof structure (which since we don't admit the Mix rule does not represent any proof in the fragment **MLL**). The question of knowing if a given structure is a proof net or not is therefore of interest. This is the search for a *criterion* that characterize proof nets. Proof nets that are also said to be *correct* proof structure. Such a criterion is therefore called *correctness criterion*.

Switching. Such criterions are known for the **MLL** fragment. The most famous one is due to Danos and Regnier [DR89], and it is the one we will use here. To express this criterion we need to introduce the notion of *switching* of a proof structure. Each $⅋$ link has two *switch* which correspond to a choice of one of the input arrow, see . We then call switching of a proof structure the graph obtained after 'switching' on every $⅋$ node of the proof structure.

More formally we can define the switching of a proof structure S, any function σ of domain $S^{(⅋)}$, that is the nodes of S labeled by $⅋$. And of codomain $Arr(S)$ the arrows of S. Such that to a node u labeled by $⅋$, the associated arrow $\sigma(u)$ is a (one of the two) input arrows of u.

Therefor it can be seen as if, for each $⅋$ node of S we can make two possible transformation. Given a switching σ of a proof structure S, we let σS denote the *graph* (it is not necessarily a proof structure) obtained as follow; the nodes of σS are $S \setminus S^{(⅋)}$. Any arrow of S, that targets a non $⅋$ node (or that has no target), is an arrow in σS. Given an arrow a that targets a $⅋$ node u, if $\sigma(u) = a$, then the target of a in σS is the target of u in S. If $\sigma(u) \neq a$ then a is a conclusion arrow in σS.

σS is called a *switching graph* of S, or a *switch* of S.

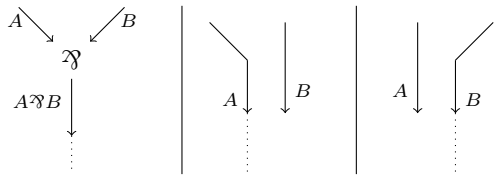


Figure 13: A ⅋ link, And its left and right switching

We then obtain the correctness criterion for **MLL** proof structure.

Definition 3.2 (Correct proof structure)**.** We say that a proof structure is *correct*, if and only if, all of its switchings are acyclic and connected.

Theorem 3.1 (Danos-Regnier correctness criterion, 1989 [DR89])**.** *A proof structure is a proof net, if and only if, it is correct.*

Cut Elimination. To get into the heart of the article of Bechet [BEC98], we need to introduce the cut elimination on proof structures. Cut elimination is a relation of arity 2 on proof structures, it is defined as in 14. We will denote the relation of cut elimination by $\rightsquigarrow$. A proof structure is said to be *normal* if one can not apply any cut elimination step on this graph – we could equivalently say that it does not contain any redex –.

Cut elimination will be said to be the *interactive* component of the proof structures. That is what we refer to in the title of this document by *good interactive behavior*.

Cut elimination of proof structures enjoys strong normalization (the number of nodes strictly reduces with one step). It is also important to state that it preserves correctness. Althought it does not preserve incorrectness.

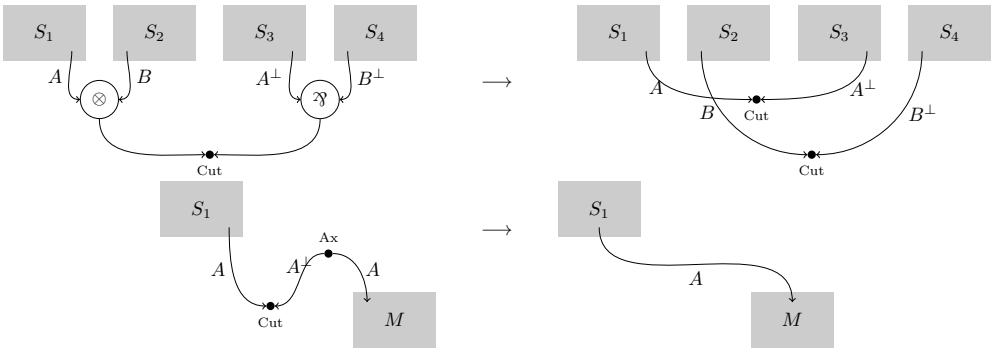


Figure 14: Cut-elimination for multiplicative proof structures. In the second redex the two arrows labeled by A of S_1 and M are supposed to be distinct.

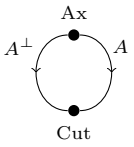


Figure 15: The deadlock proof structure.

We also want to point that, while it is true that a proof structure without cut nodes is normal, the reverse isn't true; a structure can contain cut nodes and still be normal, the canonical exemple being the *deadlock*. Note that this proof structure has no output, therefore it cannot interact (in the sense of confronting it to another structure by linking them with a cut node).

4 An analysis of proof structures through the lens graph theory

In this section we study notions of graph theory, using the notion of quivers, an object that can be found in algebra. We first give a results that allows to decompose graphs into composition of ascending and descending paths. Then we study cycles and show that any cycle can be looked at in a standard way, ie. that it has a standard representative. Finally we discuss of the meeting point of two paths, and give an upper bound on the meeting points of a family of paths. We conclude, on a result to characterize cycles, by meeting points, this is a characterization that Béchet uses and his key to his argumentation in [BEC98].

4.1 Composition and decomposition of Paths

We try to describe proof structures and modules, using the available language of quivers, an object related to algebra and graph theory, see Brion's paper [Bri08].

Definition 4.1 (Multidigraphs)**.** We call *directed multidigraph* or *quiver* a quadruplet (V, Arr, src, tgt), where;

- V and Arr are two disjoint sets.
- src and tgt are two partial[17] functions of domain Arr and of codomain V.
- The element of V are called *vertices*. While the element of Arr are called *arrows*. The function tgt is called the *target*, while src is called the *source*.

We call *unoriented multidigraph* a triplet (V, Arr, brd) where;

- V and Arr are two sets.
- brd is a total function of domain Arr and of codomain $\mathcal{P}^{1,2}(V)$, the set of subsets of V made of 1 or 2 element. brd is called the *frontier* or the *border* function.

Given a quiver (V, Arr, src, tgt), we call *labeling* a quadruplet $(\ell_V, \Sigma_V, \ell_A, \Sigma_A)$ such that:

- Σ_V and Σ_A are two disjoint sets of symbols.
- $\ell_V : V \to \Sigma_V$ and $\ell_A : Arr \to \Sigma_A$ are two total functions.

We fix some conventions ; we will make the arrows range over the symbols $\alpha, \beta, \gamma, ...$ while the vertices will range over the symbols $u, v, w, ...$

Also to avoid the heavyness of parenthesis, we might denote $\mathsf{src}.\alpha$ for $\mathsf{src}(\alpha)$ and $\mathsf{tgt}.\alpha$ for $\mathsf{tgt}(\alpha)$.

[17] In the common litterature of graph theory, the functions source and target are total. Although, proof structures (and more broadly modules) can have arrows without target nor source, this is why we make the choice to define these functions as partial functions.

Forgetting orientation, the border. Note that a directed multidigraph always induce a (canonical) unoriented one, since from the two functions tgt and src we can define a frontier function simply as, $\mathsf{brd}.brd : \alpha \to \{src.\alpha, tgt.\alpha\}$. This canonical function will be the *border* and denoted brd.

Inputs, outputs. Given a vertice v we call *inputs* or *entering arrow* of v, any arrow α such that $\mathsf{tgt}.\alpha = v$. We also call *outputs* or *outgoing arrow* of v any arrow such that $\mathsf{src}.\alpha = v$. We can distinguish some types of node;

- A node without entering arrows is called a *source* or an *introduction* node[18].
- A node without outgoing arrows is called a *sinking node* or a *well*.

Arrows as a relation on vertices. Given a quiver, and two of its nodes u and v. We denote $u \to v$ to mean that there exists an arrow α such that $\mathsf{src}.\alpha = u$ and $\mathsf{tgt}.\alpha = v$. We might also use the explicit notation $u \xrightarrow{\alpha} v$ to have a direct reference to the α, meaning that $\mathsf{src}.\alpha = u$ and $\mathsf{tgt}.\alpha = v$.

Furthermore we will denote $\to^*$ the transitive closure[19] of $\to$, note that is is not said to be reflexive. If $u \to^* v$ we will say that u *can reach* v.

Definition 4.2 (Path)**.** Given a graph (V, Arr, tgt, src). We call *path*, any finite sequence of arrows $(\alpha_1, ..., \alpha_n)$ such that, it respects a condition of *connectivity*; this means for each index i,

$$\mathsf{brd}.\alpha_i \cap \mathsf{brd}.\alpha_{i+1} \neq \emptyset. \quad \text{(is non empty)}$$

And a second condition of non redundancy or *conciseness*, meaning for each pair of unequal indexes $i \neq j$;

$$\mathsf{brd}.\alpha_i \cap \mathsf{brd}.\alpha_{i+1} \bigcap \mathsf{brd}.\alpha_j \cap \mathsf{brd}.\alpha_{j+1} = \emptyset.$$

We call *track* the set of nodes belonging to the borders of the arrows of the path. Formally;

$$Track(\rho) = \bigcup \mathsf{brd}.\alpha_i$$

We call *internal vertice* of the sequence any vertices that belongs to two borders simultaneously. Formally it correspond to the following set;

$$Int(\rho) = \{v \in V \mid \exists 1 \leq i \leq n, v \in \mathsf{brd}.\alpha_i \text{ and } v \in \mathsf{brd}.\alpha_{i+1}\}.$$

We define the *adjacence* of the path as the following;

$$Adj(\rho) = \bigcup_{1 \leq i \leq n-1} \{\mathsf{brd}.\alpha_i \cap \mathsf{brd}.\alpha_{i+1}\}.$$

[18] Think of the axiom in a proof structure.

[19] Given a binary relation R on a set X, the transitive closure of R is the relation R', such that for any $x, y \in X$: $xR'y \Leftrightarrow$ there exists $x_1, ..., x_n \in X$ such that $x = x_1 R x_2 ... R x_n = y$.

On the other we call *extremities* of the sequence, any vertice that belongs to at most one arrows border.

$$Ext(\rho) = Track(\rho) \setminus Int(\rho).$$

If the sequence is of length greater than 2, we call *source* of the path the vertice (if it exists) of $\mathsf{brd}.\alpha_1$ that is not internal. While the *target* of the path is the vertice of $\mathsf{brd}.\alpha_n$ that is not internal. [20]
The length n of the sequence is called the *length* of the path.

The set of finite sequences of arrows is denoted Arr^*. The paths, or generally the sequences of arrows, will range over the symbols $\rho, \xi, \tau, ...$

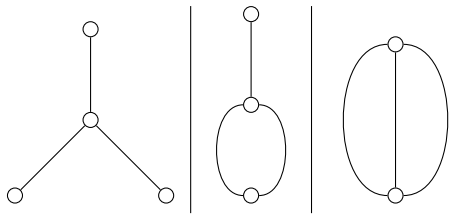


Figure 16: Some exemple of sequence respecting the condition of connectivity. We seek to reject the first figure as a notion of path, indeed it does not verify the conciseness condition.

Reachability of vertices and Paths. We want to point out that given two vertices u and v the relation of reachability $u \to^* v$ is equivalent to the existence of a descending path $(\alpha_1, ..., \alpha_n)$ from u to v. ie. such that $\mathsf{src}.\rho = u$ and $\mathsf{tgt}.\rho = v$.

Sub–paths. Given a path $(\alpha_1, ..., \alpha_{n+1})$ the subsequence $(\alpha_1, ..., \alpha_n)$ remains a path. In fact the condition of *connectivity* and of *conciseness* are both conserved since, the set of index $\{1, ..., n\}$ is obviously included in $\{1, ..., n+1\}$. But also since the pairs $(i, i+1)$ where i ranges in $\{1, ..., n\}$ are also obviously pairs where we allow i to range in $\{1, ..., n+1\}$.

The same thing holds for the same reason in the other direction, if $(\alpha_0, ..., \alpha_n)$ is a path then so is $(\alpha_1, ..., \alpha_n)$. From these two proposition we can ensure that if a sequence $(\alpha_1, ..., \alpha_n)$ is a path then so is the sub–sequence $(\alpha_{i_0}, ..., \alpha_{i_0+k})$. Where i_0 and $i_0 + k$ are in $\{1, ..., n\}$. Such a subsequence is called a sub–path of $(\alpha_1, ..., \alpha_n)$.

[20] Not that it is not ensured that a path has a source or a target, for exemple in the case of a cycle we cannot define these points.

Inversion of paths. Given a sequence of arrows $\rho = (\alpha_1, ..., \alpha_n)$, we will call the *inversion* of ρ, that we will denot by ρ^{-1} the following sequence : $\rho^{-1} = (\alpha_n, ..., \alpha_1)$.

A kind of immediate result is the following; ρ is a path if and only if, ρ^{-1} is a path. The proof is still merely based on a play of indexes.

Composition. Given two sequences of arrows ρ, τ their composition is defined as the concatenation of the sequences. We want to exhibit a *composability characterization*, given two path ρ and τ when is their concatenation a path.

Proposition 4.1 (Composability of paths)**.** *Given two paths $\rho = (\alpha_1, ..., \alpha_n)$ and $\tau = (\beta_1, ..., \beta_k)$. Denoting $I = \mathsf{brd}.\alpha_n \cap \mathsf{brd}.\beta_1$ we have the equivalency;*

$$\rho\tau \text{ is a path } \Rightarrow (I \neq \emptyset) \wedge (I \cap Int(\rho) = \emptyset) \wedge (I \cap Int(\tau) = \emptyset).$$

Proof. **(1⇒2).** Let's assume $\rho\tau$ is a path. The connectivity condition ensures that $I = \mathsf{brd}.\alpha_n \cap \mathsf{brd}.\beta_1$ is non–empty. The condition of conciseness ensures that $I = \mathsf{brd}.\alpha_n \cap \mathsf{brd}.\beta_1$ is distinct from each $\mathsf{brd}.\alpha_i \cap \mathsf{brd}.\alpha_{i+1}$, and so that $I \notin Int(\rho)$. The same thing ensures that $I \notin Int(\tau)$.

(2⇒1). Now let us assume that I is non empty and so it is not an internal element of ρ or τ. Note that the borders $\mathsf{brd}.\gamma_i \cap \mathsf{brd}.\gamma_{i+1}$ where γ_i is in the sequence $\rho\tau = (\alpha_1, ..., \alpha_n, \beta_1, ..., \beta_k)$. If both γ_i and γ_{i+1} are in the same subpath ρ and τ, then this intersection is non empty. Otherwise the intersecion corresponds to I and so by assumption it is non empty. The condition of connectivity is therefore ensured.

For the condition of conciseness. We have to ensure that I is disjoint from other borders intersection, indeed since by assumtion I does not meet $Int(\rho)$ nor $Int(\tau)$. □

Ascending and descending paths. We will distinguish two types of paths, *ascending* and *descending* paths;

- a path ρ is said to *descending* if and only if, for each index $1 \leq i \leq n$;

$$\mathsf{brd}.\alpha_i \cap \mathsf{brd}.\alpha_{i+1} = \{\mathsf{tgt}.\alpha_i, \mathsf{src}.\alpha_{i+1}\}.$$

- On the other hand ρ is said to *ascending* if and only if, for each index $1 \leq i \leq n$;

$$\mathsf{brd}.\alpha_i \cap \mathsf{brd}.\alpha_{i+1} = \{\mathsf{src}.\alpha_i, \mathsf{tgt}.\alpha_{i+1}\}.$$

Note that the target and source of a descending line corresponds to $\mathsf{tgt}.\alpha_n$ and $\mathsf{src}.\alpha_1$. While the source and target of an ascending line correspond to $\mathsf{tgt}.\alpha_1$ and $\mathsf{src}.\alpha_n$.

Proposition 4.2 (Decomposition of paths)**.** *Given a path ρ, ρ corresponds to a composition of ascending and descending paths. Meaning there exists a number k called the* number of inversion *and a function $D_\rho : \{1, .., k\} \to Arr^*$, such that;*

- $\rho = \prod_{1 \leq i \leq k} D_\rho(i)$.
- $D_\rho(1)$ *is either an ascending or a descending line.*
- *For any index* $D_\rho(i)$ *is an ascending (resp. descending) line* $\Rightarrow D_\rho(i+1)$ *is a descending (resp. ascending) line.*

Proof. By Induction. If the line is of length 1 then indeed it correspond to both a descending and ascending paths – since we consider only pairs $(i, i+1)$ –. If we assume that the proposition holds for n let's say it still holds for $n+1$. To do so we consider a path $\rho = (\alpha_1, ..., \alpha_{n+1})$, naturally consider its subsequence $\tau = (\alpha_1, ..., \alpha_n)$ we know it still correspond to a path. Therefore we can call the induction on τ and ensure that it corresponds to a composition of ascending and descending paths.

$$\tau = \prod_{1 \leq i \leq k} D_\tau(i)$$

Our aim is now to define D_ρ, we define for each $1 \leq i < k$ $D_\rho(i) = D_\tau(i)$. It remains now to define $D_\rho(k)$ and eventually $D_\rho(k+1)$.

Now note that $\rho = \tau\alpha_{n+1}$ is a path meaning this ensure the composability of the two zub–paths. So it means the following. Either $\mathsf{src}.\alpha_{n+1} \in \mathsf{brd}.\alpha_n$ or $\mathsf{tgt}.\alpha_{n+1} \in \mathsf{brd}.\alpha_n$.

But also note that either $D_\tau(k)$ is descending or ascending[21]. And also that in particular $D_\tau(k)\alpha_{n+1}$ is a path. In the end four cases can occur;

1. If $D_\tau(k)$ is descending and $\mathsf{src}.\alpha_{n+1} \in \mathsf{brd}.\alpha_n$. By assumption $\mathsf{brd}.\alpha_n \cap \mathsf{brd}.\alpha_{n+1}$ is non–empty. But note that since the sub–path is descending, $\mathsf{brd}.\alpha_{n-1} \cap \mathsf{brd}.\alpha_n = \{\mathsf{tgt}.\alpha_{n-1}, \mathsf{src}.\alpha_n\}$. This ensures that $\mathsf{tgt}.\alpha_n \in \mathsf{brd}.\alpha_{n+1}$ [22] and so $\mathsf{brd}.\alpha_n \cap \mathsf{brd}.\alpha_{n+1} = \{\mathsf{tgt}.\alpha_n, \mathsf{src}.\alpha_{n+1}\}$.

   Meaning that $D_\tau(k)\alpha_{n+1}$ is a descending path. In this case we fix $D_\rho(k) = D_\tau(k)\alpha_{n+1}$.
2. If $D_\tau(k)$ is descending and $\mathsf{tgt}.\alpha_{n+1} \in \mathsf{brd}.\alpha_n$ then merely think that $D_\rho(k) = D_\tau(k)$ is descending while $D_\rho(k+1) = \alpha_{n+1}$ is ascending.
3. If $D_\tau(k)$ is ascending and $\mathsf{tgt}.\alpha_{n+1} \in \mathsf{brd}.\alpha_n$, then $D_\tau(k)\alpha_{n+1}$ is an ascending path. This ensures that $\mathsf{brd}.\alpha_{n-1} \cap \mathsf{brd}.\alpha_n = \{\mathsf{src}.\alpha_{n-1}, \mathsf{tgt}.\alpha_n\}$, meaning in particular that $\mathsf{tgt}.\alpha_n$ is an internal point of $D_\tau(k)$.

   Since $D_\tau(k)$ and α_{n+1} are composable thanks to the last proposition, this ensure the intersection $I := \mathsf{brd}.\alpha_n \cap \mathsf{brd}.\alpha_{n+1}$ is non empty, but also that it is disjoint from the internal point of $D_\tau(k)$, therefore it can only be that $\mathsf{src}.\alpha_n$ is in I. And so $I = \{\mathsf{src}.\alpha_n, \mathsf{tgt}.\alpha_{n+1}\}$.

   Recording that $D_\tau(k)$ is ascending, this ensure that $D_\tau(k)\alpha_{n+1}$ is descending. In this case we fix $D_\rho(k) = D_\tau(k)\alpha_{n+1}$
4. If $D_\tau(k)$ is ascending and $\mathsf{src}.\alpha_{n+1} \in \mathsf{brd}.\alpha_n$, then $D_\tau(k) = D_\rho(k)$ is ascending while $D_\rho(k+1) = \alpha_{n+1}$ is descending.

[21] $D_\tau(k)$ is necessarily of the form $(\alpha_{i_0}, ..., \alpha_n)$.

[22] Since either the target or the source of α_n is in the borders of α_n and α_{n+1}, but also thanks to conciseness, indeed if the source of α_n was also in the border of α_{n+1}, then an internal point of $D_\tau(n)$ is in the intersection $\mathsf{brd}.\alpha_n \cap \mathsf{brd}.\alpha_{n+1}$, which goes against composability thanks to the last proposition.

Then simply remark that in the case (1) and (3) it corresponds to k concatenation while in the case (2) and (4) it corresponds to $k+1$ concatenation with $D_\rho(k+1) = \alpha_{n+1}$,

$$\rho = \prod_{1 \leq i \leq k} D_\rho(i) \quad \text{or} \quad \rho = \prod_{1 \leq i \leq k+1} D_\rho(i).$$

□

Forms of a path. Note that this last proposition on decomposition ensures that a path ρ is the concatenation of successively ascending and descending path, but therefore it is always in one of the four forms;

1. $\rho = \prod_{1 \leq i \leq n} \delta_i \iota_i$
2. $\rho = (\prod_{1 \leq i \leq n} \delta_i \iota_i)\delta_{n+1}$
3. $\rho = \prod_{1 \leq i \leq n} \iota_i \delta_i$
4. $\rho = (\prod_{1 \leq i \leq n} \iota_i \delta_i)\iota_{n+1}$

Where $(\delta_i)_{1 \leq i \leq n+1}$ and $(\iota_i)_{1 \leq i \leq n+1}$ are two sequences of respectively descending and ascending path that are composable.

The first case (1) corresponds to the case where the number of inversion is pair and the first sub–path $D_\rho(1)$ is descending. The second case corresponds to a first sub–path that is descending and an even number of inversion. And so on for the two other cases.

4.2 Standard cycle Representative in a strongly oriented quiver

Definition 4.3 (Cycle)**.** A *cycle* is a path $(\alpha_1, ..., \alpha_n)$ such we also have the conditions of connection and conciseness extended to the ordered pair of indexes $(n, 1)$;

$$\mathsf{brd}.\alpha_n \cap \mathsf{brd}.\alpha_1 \neq \emptyset. \quad \text{(is non empty)}$$

$$\mathsf{brd}.\alpha_n \cap \mathsf{brd}.\alpha_1 \bigcap \mathsf{brd}.\alpha_i \cap \mathsf{brd}.\alpha_{i+1} = \emptyset$$

Where i ranges over $\{1, ..., n\}$. We might call n–cycle, a cycle of length n.

Standard cycles. Note that a cycle is also a path, therefore it also in one of the given forms. A cycle in the form (1) ie. that start by a descending path and as an even number of inversion, is said to be *standard* or *in standard form*.

One of our goal is to reduce the study of cycles, to the study of cycles of the form (1) with enough specification on the given quiver (that are coherent with proof nets), we can ensure that any cycles corresponds to a cycle in standard form.

Cycles Equivalency. A cycle is a fortiori a sequence $(\alpha_1, ..., \alpha_n)$, we want to stress that the cycles enumeration is more flexible than the one for a paths. We mean that any sequence $(\alpha_i, \alpha_{i+1}, ..., \alpha_n, \alpha_1, ..., \alpha_{i-1})$ will equivalently be a cycle. The proof is – again – based on a play on the indexes.

Given two cycles C and C' we will say they are equivalent, if and only if, $C = (\alpha_1, ..., \alpha_n)$ while C' is a transposition of C ie. of the form $C' = (\alpha_i, \alpha_{i+1}, ..., \alpha_n, \alpha_1, ..., \alpha_{i-1})$. We might then denote $C \equiv C'$.

Proposition 4.3 (Cycles equivalency)**.** *Given a non directed cycle* $(\alpha_1, ..., \alpha_n)$ *then* $(\alpha_2, ..., \alpha_n, \alpha_1)$ *is also cycle.*

And so it directly follows that any sequence $(\alpha_{i+1}, ..., \alpha_n, \alpha_1, ..., \alpha_i)$ *is a also a cycle.*

Proof. The proof is merely based on a play on indexes. Note that the condition of connectivity and conciseness of a cycle $(\alpha_1, ..., \alpha_n)$ are operating on pairs of the form $(i, s(i))$ where i is the successor permutation of $\{1, ..., n\}$. Indeed these pairs are the same in $(\alpha_2, ..., \alpha_n, \alpha_1)$. □

Now our goal is to specify conditions on quivers that are coherent with proofnets, and such that any cycles C is equivalent to a cycle C_0 that is in standard form. The first obvious thing is to require that our graph as no *directed cycle*, that is a cycle that is an ascending or a descending path.

Proposition 4.4 (Strongly Oriented quiver)**.** *Given some quiver* $G = (V, Arr, src, tgt)$*, the two assertions are equivalent;*

1. *The relation* $\to^*$ *is (strict and partial) ordering of* V*.*
2. *G contains no directed cycle.*

A quiver satisfying one of these (equivalent) statement will be said to be strongly oriented. We choose this terminology since in the common litterature an oriented graph is a graph with no 2 *cycles.*

Proof. First remark that $\to^*$ is a transitive relation. Indeed if $u \to v$ and $v \to w$, it means there are two descending path $(\alpha_1, ..., \alpha_n)$ and $(\beta_1, ..., \beta_k)$ such that respectively, $src.\alpha_1 = u$, $tgt.\alpha_n = src.\beta_1 = v$ and finally $tgt.\beta_k = w$. Indeed then considering the sequence obtained by their composition (or we could say concatenation), $(\alpha_1, ..., \alpha_n, \beta_1, ..., \beta_k)$ is first a descending path. But also it a path from u to w, this shows that $u \to^* w$

Now let us show the equivalency.

(1⇒2). Assume $\to^*$ is strict and partial ordering of V. Then G cannot contain any directed cycle, which by definition would be a path ρ from some vertices v to v. This would mean $v \to^* v$. This would go against v being anti reflexive.

(2⇒1). Assume that G has no directed cycles, this shows that $\to^*$ is antireflexive. It also ensures antisymmetry indeed, assume that $u \to^* v$ and $v \to^* u$, for some vertices of G.

Note that we know $\to^*$ to be transitive, therefore we would, $u \to^* u$. Which exactly means there exists a descending path from u to u, ie. a cycle. That would be a contradiction. □

Proposition 4.5 (Sufficient condition for standard form)**.** *Given a non directed cycle $C = (\alpha_1, ..., \alpha_n)$ and the graph G contains no directed cycle.*

If the vertice $\mathsf{src}.\alpha_1$ *is a source for the induced graph by C, then C is a cycle in* standard form.

Proof. Indeed C being a cycle $brd.\alpha_1 = \{v_1, v_2\}$ and $brd.\alpha_n = \{v_n, v_1\}$. Since v_1 has no entering arrows from $\{\alpha_1, ..., \alpha_n\}$ these egalities ensure that $\mathsf{src}.\alpha_1 = v_1$ and $\mathsf{src}.\alpha_n = v_1$.

The decomposition of C therefore starts by a descending path and ends by an ascending path. Meaning C is in standard form. (form (a) in the proposition **??**) □

Proposition 4.6. *In a strongly oriented quiver, any cycle contains at least one node with at least* 2 *outgoing arrow.*

Proof. Since the quiver is supposed to be strongly oriented, it cannot be that the cycle C is an ascending or descending path, it is a composition of ascending and descending path, so it contains two subpath ρ that is ascending and τ that is descending, so that $\rho\tau$ is a path. The point contained in $\mathsf{brd}.\alpha_n \cap \mathsf{brd}.\beta_1$ is a point with two outgoing arrow. □

Sequential graph. Note that this two last proposition, can explicit a condition on quivers such that, we can ensure the existence of a canonical form for any cycles. Since we can always ensure the existence of a node with 2 outputs (or more) from the proposition 4.6. And that with the proposition 4.5 we see that having a node with no input is a sufficient condition to have a canonical form.

These two proposition explicit a condition on quivers that ensure the existence of canonical forms. In a straightforward way the condition would merely be the following ; Given any vertice v, if v has more than 2 output then v has no input.

Note that indeed this is something that is true in proof nets, in fact it comes from the fact that the rules (except for the axiom rule), result in at most one formula either by introducing a connector, or by lowering the number of formulas.

The condition can be formulated, given any vertices if v has some inputs, then v has at most 1 output. The idea behind this is that the operation on the sequents that result from a rule, are enough decomposed so that we can follow the process of the proof, ie. there are no two operations being done simultaneously. For this intuition we decide to call a quiver respecting this condition a *sequential graph* or *sequential quiver*.

Proposition 4.7 (Existence of cycles canonical representative)**.** *Given a strongly oriented quiver that is also sequential.*

Any cycle has a standard representative.

Proof. Given a cycle $C = (\alpha_1, ..., \alpha_m)$, we can ensure that C contains a node with two output v calling the proposition 4.6, therefore since the graph is sequential this node has no input. From there calling the proposition 4.5 we can ensure that there exists a cycle $C_0 \equiv C$ starting from the node v that is in canonical form. □

4.3 Tracks and meeting points

In this subsection we assume that we are given a quiver $G = (V, Arr, tgt, src)$ that is strongly oriented. We define the notion of track, and show that tracks are well ordered sets – with respesct to $\to^*$ – in a strongly oriented quiver. Based on the notion of track, we then define the notion of meeting point, we then show that, in what will be said to be a binary quiver, the number of meeting points can be bounded, this corresponds to the result of the proposition 4.10. This upper bound is the key for us to show the equivalency presented in the lemma 9.1.

In a strongly oriented quiver, given two vertices u and v, we might say that;
- u is *below* or *under* v, if there exists an ascending path from u to v.
- u is *above* v, if there exists a descending path from v to u.

We can easily note that it is equivalent that u is below v and v is above u.[23]

Definition 4.4 (Track)**.** We call track of a path $\rho = (\alpha_1, ..., \alpha_n)$ the set of vertices denoted $track(\rho)$ of points in the border of any arrows α_i;

$$track(\rho) = \bigcup_{1 \leq i \leq n} \mathsf{brd}.\alpha_i$$

Proposition 4.8 (Tracks are well ordered sets)**.** *Given $\rho = (\alpha_1, ..., \alpha_n)$ a descending path, $track(\rho)$ is a totally ordered set by $\to^*$.*

Proof. We can do so by induction on the length of the path. Assume $\rho = \alpha_1$ is made of one arrow; then $track(\rho) = \{src.\alpha_1, tgt.\alpha_1\}$. Indeed we have $src.\alpha_1 \to tgt.\alpha_1$ and therefore $src.\alpha_1 \to^* tgt.\alpha_1$.

Now let us assume ρ is made of $n + 1$ arrow, consider it's first n arrow and apply induction. Now take u, v two arrows of ρ and let's show they are comparable by $\to^*$.

- If u, v are both in the border of some – eventually distinct – α_i with $1 \leq i \leq n$, then they are in the track of $\rho_0 = (\alpha_1, ..., \alpha_n)$ therefore by induction they are comparable by $\to^*$.
- If u, v are both in the border of α_{n+1}, then they are either equal and so comparable. Or $u = src(\alpha_{n+1})$ while $v = tgt(\alpha_{n+1})$ (eventually inverting) and so $u \to^* v$.

[23] Given an ascending path from u to v, it's inversion is a descending path from v to u.

- Assume u is in the border of an α_i of ρ_0 while $v = src.\alpha_{n+1}$. Note that since ρ is a direct path $tgt(\alpha_n) = src(\alpha_{n+1})$, but notice that $tgt.\alpha_n$ is in the track of ρ_0, so it follows that $u \to^* tgt.\alpha_n = v$.
- Finally if u is in the border of an α_i of ρ_0 while $v = tgt.\alpha_{n+1}$. from the previous point $u \to^* tgt.\alpha_n$ and $tgt._n \to tgt.\alpha_{n+1}$. So we can conclude $u \to^* tgt.\alpha_{n+1} = v$.

We treated the possible cases, ensuring that any two points of $track(\rho)$ are comparable by $\to^*$. Meaning $track(\rho)$ is well ordered by $\to^*$. □

Definition 4.5 (Meeting point)**.** In a strongly oriented quiver we can define the notion of meeting point.

Given two paths ρ, ξ we call meeting point of ρ and ξ, that we denote $\rho \wedge \xi$, the smallest element with respect to $\to^*$ of $track(\rho) \cap track(\xi)$.

Proposition 4.9 (Extensions preserves meeting points)**.** *Given ρ and ξ two paths, if ρ and ξ meet in some vertice v.*

Then given any extension of ρ, $\tau = \rho\rho'$, τ and ξ meet in m.

Proof. Note that since $\tau = \rho\rho'$ it naturally follows that $track(\rho) \subset track(\tau)$.

Now by hypothesis m is the smallest element of $track(\rho) \cap track(\xi)$. Let's show that it is also the smallest element of $track(\tau) \cap track(\xi)$. Consider a vertice v in the track of τ and ξ. v is in the track of $\rho\rho'$ so it is either in the track of ρ' or in the track of ρ.

If v is in the track of ρ by hypothesis indeed, $m \leq v$. If v is in the track of ρ' indeed given any $u \in track(\rho)$ we have $u \xrightarrow{\rho\rho_0} v$ and so $u \leq v$.

Now m is (a fortiori) an element of the track of ρ so it follows that $m \leq v$. □

Binary Quiver. We will say that a quiver is binary when, all its node have at most 2 inputs. This condition is key to give an upper bound to the number of meeting points, of a family paths. This upper bound will later be of use to study the cyclicity in proof structures.

Lemma 4.1. *Given a binary quiver. Given three paths ρ, ξ, τ.*

Assume that $\rho \wedge \xi \to^* \rho \wedge \tau$. Then it follows that the meeting points are equal, $\tau \wedge \xi = \rho \wedge \tau$.

Proof. Let's assume that $\rho \wedge \xi \to^* \rho \wedge \tau$. Note that $\rho \wedge \tau$ is then in the track of ρ, ξ, τ. Therefor we can ensure that $\tau \wedge \xi \leq \rho \wedge \tau$.

$\rho \wedge \tau$ is by definition the target of two subpaths $\rho_0 \neq \tau_0$ that are diverse. On the other hand $\rho \wedge \xi \to^* \rho \wedge \tau$, ensure two paths $\rho_0 \neq \xi_0$ of target $\rho \wedge \tau$.

Ad absurdum, assuming that $\tau \wedge \xi < \rho \wedge \tau$ meaning that $\tau \wedge \xi \to^* \rho \wedge \tau$. This means there exists two paths $\tau_0 \neq \xi_0$ of target $\rho \wedge \tau$.

We therefore have three distinct path ρ_0, ξ_0, τ_0 of target $\rho \wedge \tau$ this would mean that this vertice has three input, which is supposed to be impossible in a binary quiver. □

Proposition 4.10 (Meeting points bounding)**.** *Given a binary input quiver, and given* $(\rho_1, ..., \rho_n)$ *a family of descending paths. Denoting* m_i *the meeting point of* ρ_i, ρ_{i+1}*, and* m_n *the meeting point of* ρ_n, ρ_1*.*

Then the set of meeting points $\{m_1, ..., m_n\}$ *is made of at most* $n-1$ *nodes.*

Proof. If, $n = 2$ ok. By induction now consider $(\rho_1, ..., \rho_{n+1})$ a sequence of paths. Assume, all m_i are distinct. The induction on the family $(\rho_1, ..., \rho_n)$ ensures that $\rho_n \wedge \rho_1$ is equal to some m_k.

$\rho_{n+1} \wedge \rho_1$ and $\rho_n \wedge \rho_{n+1}$ are both point of the track of ρ_{n+1} they are therefore comparable, let's operate by cases

- If they are equal then indeed $m_1, ..., m_{n+1}$ is made of n nodes at most, and we are done.
- If $\rho_{n+1} \wedge \rho_1 \to^* \rho_n \wedge \rho_{n+1}$ then calling the previous lemma, $\rho_{n+1} \wedge \rho_1 = \rho_n \wedge \rho_1$, but, by induction $\rho_n \wedge \rho_1$ is redundant.
- If on the other hand. $\rho_{n+1} \wedge \rho_n \to^* \rho_{n+1} \wedge \rho_1$ then calling the last lemma again $\rho_{n+1} \wedge \rho_n = \rho_n \wedge \rho_1$ but, by induction $\rho_n \wedge \rho_1$ is redundant.

□

Arrangement, Focused Quiver. Given a quiver (V, A, src, tgt). Given an arrow α, the set of paths starting at α is called the arrangment of α. It is denoted Arr^*_α, and so is defined as the following set;

$$Arr^*_\alpha = \{\rho \in Arr^* \mid \alpha \prec \rho\}$$

.

We say that quiver is *focused* if for any arrow α its arrangement Arr^*_α is totally ordered by $\prec$.

Note that since the number of arrows of a quiver is supposed to be finite, in a focused quiver it follows that, there is a greater path starting by α, for any arrows α. The greatest path starting with α might be called the *realisation* of α and will be denoted $rl(\alpha)$.

The meeting points of two arrows α, β can then be defined in such a quiver; as the meeting point of their two realisation path, $rl(\alpha)$ and $rl(\beta)$. It will be denoted $\alpha \wedge \beta$.

Corollary 4.0.1. *Given* ρ *and* ξ *two paths that meets, and their first arrow being respectively* ρ_0 *and* ξ_0*.*

In a focused quiver; if ρ *and* ξ *meet, then* $\rho \wedge \xi = \rho_0 \wedge \xi_0$*.*

Proof. Note that ρ is in the set of the paths starting by ρ_0. Being in a focused quiver, there exists such a path that is maximal let's denote it ρ', and so by definition $\rho \leq \rho'$.

Calling the proposition 4.9 it follows that $\rho \wedge \xi = \rho' \wedge \xi$.

Now note that ξ is a path starting by ξ_0, and therefore for the same reasons, $\xi \leq \xi'$. Still calling the proposition **??** it follows that $\rho \wedge \xi = \rho' \wedge \xi = \rho' \wedge \xi'$. □

Proposition 4.11. *Given two arrows α, β there exists a smallest path starting by α ie. of Arr^*_α containing the meeting point $\alpha \wedge \beta$ in its track.*

We call this path the meeting path of α and β. We denote the meeting path by it $p(\alpha, \beta)$,

Proof. Consider α and β two arrows, let's assume that these two arrows meet in $v = \alpha \wedge \beta$. This means that the set $\{\rho \in Arr^*_\alpha \mid v \in track(\rho)\}$ is non empty. But note that it is also a subset of Arr^*_α, and that by assumption the quiver is focused, meaning this set is well ordered.

This ensures that there exists a smallest path ρ of Arr^*_α containing v. Note that this smallest path necessarily targets the vertice v, otherwise it would go against the minimality of ρ in Arr^*_α. □

Meeting paths composability. In a focused quiver, given two meeting arrows α, β. The paths $p(\alpha, \beta)$ and $p(\beta, \alpha)$ are composable such that $p(\alpha, \beta)p(\beta, \alpha)^{-1}$ is a path. This is guaranteed thanks to the proposition **??**, since $p(\alpha, \beta)$ and $p(\beta, \alpha)$ both have the same target, that is the meeting point $\alpha \wedge \beta$

Finally we note that the two properties of sequentiality and focusness of a quiver are in the end equivalent, which was at first not obvious.

Proposition 4.12. *Given some quiver G the two statement are equivalent;*

1. *G is focused.*
2. *G is sequential.*

Proof. **(1⇒2).** Assume that G is focused and let us show sequentiality. Meaning that we want to show that any node with one input (or more), has at most one output. Indeed take v some vertices with one input α. assume v has two output β, γ. Then the two sequences (α, β) and (α, γ) are both path starting with α yet none of them is a subsequence of the other. We have a contradiction.

(2⇒1). If G is sequential. Consider α some arrows, and ρ and τ two path starting by α assume without loss of generality that ρ is of smaller length than τ. By induction on the length of ρ let's show they are comparable. If ρ is of length 1 indeed $\rho = (\alpha)$ on the other hand τ is some sequence starting by α therefore, $\rho \prec \tau$.

Now assume ρ is of length $n + 1$.Note that by induction $(\alpha_1, ..., \alpha_n) \prec \tau$. Therefore the identity of α_{n+1} and β_{n+1} all depends on if $tgt.\alpha_n$ has two outputs, this node has an input so since the graph is sequential it has only one conclusion, therefore $\alpha_{n+1} = \beta_{n+1}$. From this follows that $\rho \prec \tau$ □

Local composability of a concatenation. Given some concatenation of paths $\rho_1 ... \rho_n$, note that in order to ensure that this concatenation is a path, it

is sufficent to show that ρ_i and ρ_{i+1} are composable. Since if it holds induction would proove that $\rho_1\rho_2$ is a path, then so is $\rho_1\rho_2\rho_3$ and so on...

With the formalism introduced, we now present a generalized version of Bechet switching cycle argument in the article [BEC98]. Furthermore the introduced formalism will help us to obtain proofs for the construction of an acyclic proof structure from a cyclic one.

Theorem 4.1 (Bechet's Cyclicity Criterion)**.** *Given a quiver G strongly oriented and sequential. The two statements are equivalent;*

1. *There exists n introduction node*[24] *$N_1, ..., N_n$ and two sequences of arrows of the same length n, $\alpha_1, ..., \alpha_n$ and $\beta_1, ..., \beta_n$ such that*
   (a) *(Common Source). For each i, both arrows α_i and β_i are distinct and have their source in N_i.*
   (b) *(Meeting arrows). For each i, the two arrows $\alpha_i, \beta_{s(i)}$ meet.*
   (c) *(No Redundancy). The meeting points $\alpha_i \wedge \beta_{s(i)}$ are pairwise distinct.*
2. *G contains a cycle.*

here the function $s : \{1, ..., n\} \to \{1, ..., n\}$ is the permutation sends $i \neq n$ to $i + 1$ and n to 1.

Proof. **(1⇒2).** under these assumptions, let's show that we can construct a cycle. We define two sequences of paths, given an index $1 \leq i \leq n$, we define $\delta_i = p(\alpha_i, \beta_{s(i)})$ where $s(i)$ is the permutation of $\{1...n\}$ that sends n to 1, and each other integer to its successor. We define a second sequence as $\delta'_i = p(\beta_{s(i)}, \alpha_i)$

Now we want to show that

$$C = \prod_{1 \leq i \leq n-1} \delta_i {\delta'_i}^{-1}$$

Is the cycle we look for.

To do so let's first show it is a path, which is in fact most of the proof's work. To do so we base ourself on the idea that a concatenation of path is also path, if it corresponds to *'local compositions'*, ie. the $i-th$ path of the concatenation is composable with the $i+1-th$ path. So in our case we have to show that δ_i is composable with ${\delta'_i}^{-1}$ and ${\delta'_i}^{-1}$ is composable with δ_{i+1}.

- Note that given some index $1 \leq i < n$, $\delta_i = p(\alpha_i, \beta_{s(i)})$ and ${\delta'_i}^{-1} = p(\beta_{s(i)}, \alpha_i)^{-1}$ are composable – thanks to the proposition **??**, see the previous remark –.

[24] Meaning nodes with no inputs. That is a node such that it is the target of no arrow.

- Also note that $p(\beta_{s(i)}, \alpha_i)$ has for source $src.\beta_{i+1}$ and so by assumption its source is N_{i+1}. But at the same time $p(\alpha_{i+1}, \beta_{i+2})$ has for source the source of α_{i+1}, ie. by the same assumption, the source of this path is N_{i+1}.

  Therefore ${\delta'_i}^{-1} = p(\beta_{i+1}, \alpha_i)^{-1}$ and $\delta_{i+1} = p(\alpha_{i+1}, \beta_{i+2})$ are composable paths, again by calling the proposition **??**.

It is simple then to ensure that it is even a cycle since by assumption, $src.C = src.\alpha_1 = N_1$ and $tgt.C = tgt.p(\beta_1, \alpha_n)^{-1} = src.p(\beta_1, \alpha_n) = src.\beta_1 = N_1$.
(2⇒1). Assume G contains a cycle C. Note that since G is sequential and strongly oriented, this cycle is equivalent to a canonical one C_0 calling the proposition 4.7. C_0 is a canonical cycle meaning it is of the form

$$C_0 = \prod_{1 \leq i \leq n} \delta_i {\delta'_i}^{-1}.$$

where the δ_i, δ'_i are descending paths.

Now consider the sequence of nodes $(N_i)_{1 \leq i \leq n}$ defined as, for any i, $N_i = src.\delta_i$ the source of the descending path δ_i. Consider the two sequences of arrows $\alpha_i = \delta_i(1)$ is first arrow of the sequence δ_i. And $\beta_i = \delta'_{pred(i)}(1)$[25] is the first arrow of $\delta'_{pred(i)}$.

Let us remark that, given an index i, $N_i = src.\delta_i = src.\delta_i(1) = src.\alpha_i$ Note that since C_0 is a fortiori a path, ${\delta'_i}^{-1}$ has to be composable with δ_{i+1}. Therefore calling again the proposition **??** it is necessary that $src.\delta'_i = src.\delta_{i+1}$ so it follows that $src.\delta'_i = N_{i+1}$ and so $src.\delta'_{pred(i+1)} = N_{i+1}$ meaning $src.\beta_{i+1} = N_{i+1}$. Also since C_0 is a cycle $N_1 = src.\delta_1 = src.\delta'_n = src.\delta'_{pred(1)}$ showing $N_1 = src.\alpha_1 = src.\beta_1$. This means $src.\alpha_i = src.\beta_i = N_i$ for each i.

Indeed C_0 contains the targets of its direct subpath, meaning that it contains $tgt.\delta_i = tgt.\delta'_i$ [26] meaning it contains the meeting points $\delta_i \wedge \delta'_i$. This meeting points corresponds to $\alpha_i \wedge \beta_{i+1}$ and $\alpha_n \wedge \beta_1$, this ensures that these arrows meet.

Now note that necessarily all these meeting points are disctinct since none of the meeting points is the source nor the target of C_0, and since cycles are defined has having non repeating tracks. □

[25] The predecessor is here intended to be the permutation of $\{1, ..., n\}$ that sends $i+1$ to i, and 1 to n.

[26] the equality holds, since by assumption the two paths are composable. This is again a reference to the proposition **??**

5 Modules as Quivers

5.1 Modules, Proof structures and Switchings

Relating Quivers and Proof structures. Let us stress that a proof structures of **MLL** is a quiver that is strongly oriented, sequential and binary. But even better, this is also true for the proof structures of **MELL**, this could give hope for a generalization of the 'minimality of the correctness criterion'.

Definition 5.1 (Pre–module)**.** We say that a quiver (V, Arr, tgt, src) is a *pre–module* if and only if, it contains only three types of node;

- Introduction nodes with no input and 2 output.
- Sinking nodes with 2 inputs and no outputs.
- Connector nodes with 2 inputs and 1 outputs.

Then we call *pre–proof–structure* a pre–module such that its source function src is total.

Pre–modules and strong orientation. One can see that this types of quiver is naturally binary – no nodes has more than 2 input –, it is also sequential – no nodes with input has more than 1 output –. Althought without any conditions on labels, the described quiver is not necessarily strongly oriented, it could that there are oriented cycles – in particular loops – in such a quiver.

Modules. A *module* (resp. proof structure) is a pre–module (resp. pre–proof–structure) that is labeled by $(\mathcal{MLL}, \{\otimes, \parr, \mathbf{Ax}, \mathbf{Cut}\}, \ell_V, \ell_A)$ and to which we associate a function $h : Arr \to \{1, 2\}$ such that;

- Introduction nodes are labeled by **Ax**.
- Sinking nodes are labeled by **Cut**.
- Connector nodes are labeled by $\otimes$ of $\parr$.
- h is defined such that given two arrows α, β having the same target, $h(\alpha) \neq h(\beta)$. This function has use in cut elimination.

And arrows are labeled in the following way;

- Given two arrows α, β of source u a node labeled by axiom, if α is labeled by A, then β is labeled by $A^\perp$.
- Given two arrows α, β of target u a node labeled by cut, if α is labeled by A, then β is labeled by $A^\perp$.
- Given two arrows α, β of target u a node labeled by $\otimes$ (resp. $\parr$), if α is labeled by A, and β is labeled by B. And if $h(\alpha) = 1$ and $h(\beta) = 2$, then the unique arrow of source u is labeled by $A \otimes B$ (resp. $A \parr B$).

The enumerated conditions on the labels, ensures that there is no descending cycles in such a graph. Therefore a proof structure is also strongly oriented.

Notations. Given a module $\mathcal{M} = (V, Arr, tgt, src)$ we will denote by $V^{(\square)}$ the set of nodes in V that are labeled by $\square$, ie. such that $\ell(v) = \square$. Where $\square \in \{Ax, Cut, \otimes, \parr\}$.

Forgetting orientation. Given a module $\mathcal{M}$ (and so it also holds for proof structures) we might denote by $\mathcal{M}^u$ the graph obtained by forgetting the orientation.

Proof structures and Modules. The notion of *module* is a richer notion than the one of structures, in the sense that a structure is always a module, the one difference is that a module may have arrows that have no source. Such arrows are called *premisse* or *input*. While arrows with no target are called *conclusions* or *output*. Two arrows are said to be *compatible* if they are respectively labeled by A and $A^\perp$.

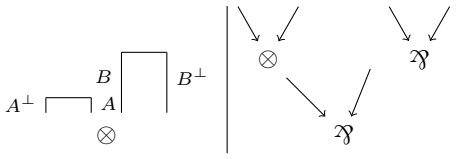


Figure 17: A proof structure and a module.

Switching Path. To have a look at switchings, we follow the definition used by Nguyên in his article [Ngu20]. Given a proof structure S we define a set $\mathcal{P}_{\parr}$ whose elements are subsets of Arr – The set of arrows of S –, precisely $\mathcal{P}_{\parr}$ is made of the pairs of arrows $\{\alpha, \beta\}$ such that $tgt.\alpha = tgt.\beta$ and the vertice $u = tgt.\alpha$ is labeled by $\parr$.

We then call a *switching path* any path ρ of S such that, for any $P \in \mathcal{P}_{\parr}$ ρ contains at most one arrow of P. And so switching cycle is merely a switching that is also a cycle.

We can now show that the meeting point of a switching cycle as described in the theorem 4.1, are necessarily labeled by tensor's $\otimes$ or cuts.

It comes from the fact that a path of the form $\delta\delta'$ where δ is descending and δ' is ascending, necessarily contains the two arrows that target the meeting point $\delta \wedge \delta'$. If this meeting point is labeled by a $\parr$, then $\delta\delta'$ does not correponds to a switching path, since it contains two arrows targeting the same $\parr$ node.

Correctness, switching, for class of modules. We introduced correctness in the first part but here we present it using the formal notions introduced. We

will also extend the correctness criterion to modules. Record that a proof–net is a proof structure that represents a formula according to the figure 12. Then we define the *correctness* as the fact that a proof structure for any of its switching the graph σS is acyclic and connected. Let us introduce the necessary notions for the class of graphs that are modules.

Definition 5.2 (Switching)**.** Given a proof structure $\mathcal{M}$. We call *switching* of $\mathcal{M}$ any function $\sigma : V^{⅋} \to Arr$ such that for any node u labeled by $⅋$, $\sigma(u)$ is one of the two entering arrow of the node u.

Given a switching σ of a module $\mathcal{M}$. We denote by $\sigma\mathcal{M}$ and call *switched graph* the pre–module obtained as the following;

- $V_{\sigma\mathcal{M}} = V_{\mathcal{M}} \setminus V_{\mathcal{M}}^{(⅋)}$. The vertices of the switched graph are the vertices of $\mathcal{M}$ that are not labeled by $⅋$.
- $Arr_{\sigma\mathcal{M}} = Arr_{\mathcal{M}} \setminus \{\alpha \in Arr_{\mathcal{M}} \mid \mathsf{src}.\alpha = ⅋\}$, the arrows are the same after and before a switch.
- $src_{\sigma\mathcal{M}} = src_{\mathcal{M}}$ the arrows have their source unchanged.
- Given an arrow α, if $u = tgt_{\mathcal{M}}(\alpha)$ is a node not labeled by a parr, then $tgt_{\mathcal{M}}(\alpha) = tgt_{\sigma\mathcal{M}}(\alpha)$. If u is labeled by parr, σ is defined on u, and if $\sigma(u) = \alpha$, $tgt_{\mathcal{M}}(\alpha_0) = tgt_{\sigma\mathcal{M}}(\alpha)$ otherwise $tgt_{\sigma\mathcal{M}}(\alpha) = \perp$ is undefined.

Cut–free switching. We want to stress that the switch of a module without cut and with one conclusion remains a module. This implies, in other terms, that the switch of tree $\mathcal{T}$ made only of $⅋$ and $\otimes$ node, is a module.

Correctness for proof structures. We then define the correctness criterion for proof structures. We will say that a proof structure is *correct* if and only if given any of its switching σ, the graph σS is acyclic and connected.

Proposition 5.1 (DR–Correctness criterion)**.** *Given a proof structure S, the tree statement are the same;*

- *S is a proof net.*
- *S is DR–correct. Meaning that for any switching, σS is acyclic and connected [DR89].*
- *S has no switching path. (in the sense of [Ngu20]). And any of its swithing is connected.*

Proof. One can look [DR89]. □

5.2 Modules Correctness, and Completion Theorem

Correctness for Modules. Now we want to generalize the correctness problem to modules. First the aim of this criterion is to identify modules that are submodules[27] of a proof–net. Again the search for a criterion is necessary,

[27] That is in the sense of isomorphism.

since the class of graphs that are modules, is strictly containing the class of sub–modules of proof–nets.

We call the frontier of a module $\mathcal{M}$ the set of input and output arrow.

$$Frontier() = Input(\mathcal{M}) \cup Output(\mathcal{M})$$

where;

$$Input(\mathcal{M}) = \{\alpha \in Arr \mid \alpha = \bot\}$$

$$Output(\mathcal{M}) = \{\alpha \in Arr \mid \alpha = \bot\}$$

We will say that a module $\mathcal{M}$ is $DR^- - correct if and only if, given any switching \sigma of M$,
$\sigma\mathcal{M}$ *is an acyclic graph.*
$\sigma\mathcal{M}$ *has no* internal component*. This means that any node of the graph is connected to the frontier.*

Given a module $\mathcal{M}$ we can –canonically – complete it in a proof structure that we will denote $\mathcal{M}^+$, by adding an axiom on each entering arrow.

Proposition 5.2. *Given any module $\mathcal{M}$ we have the equivalency;*

$$\mathcal{M} \text{ is } DR^- \text{ correct} \Leftrightarrow \mathcal{M}^+ \text{ is } DR^- \text{ correct.}$$

Proof. It is kind of straightforward, first notice that a switching of $\mathcal{M}$ is a switching of $\mathcal{M}^+$ and vice–versa.

Now take a point in $\mathcal{M}$ and consider a switching $\sigma\mathcal{M}$, since there are no internal component two case can occur. Either this point is connected to a conclusion in $\mathcal{M}$ and this remains true in $\mathcal{M}^+$. If otherwise the point is connected to an input α of $\mathcal{M}$, then in $\mathcal{M}^+$ this arrow α becomes the outgoing arrow of an axiom, axiom that is linked to a conclusion $\alpha^\bot$.

On the other diredction taking a point of $\mathcal{M}^+$ and one of its switching. Notice that if the point is connected to a conclusion of $\mathcal{M}$ this remains true. If it is connected to a conclusion α that is not a conclusion of $\mathcal{M}$ then it is connected to $\alpha^\bot$ an input of $\mathcal{M}$. □

Definition 5.3 (Saturated Module)**.** We will say that a module $\mathcal{M}$ is *saturated* if and only if, for any ⅋ link of the module, the premisses of that link are connected by one switching path – eventually more –.

Proposition 5.3 (Sufficient condition for Saturation)**.** *Given a proof structure S that is DR^- correct. If for any conclusion α and β of S there is – at least – one switching path from α to β, then S is saturated.*

Proof. Consider any ⅋ link of S, denoting its premisses p_1, p_2 and it's conclusion q. Consider a switching σ of S. In σS if p_1 and p_2 are connected, we are done.

If not notice that since S is DR^- correct, p_1 and p_2 are both connected to the frontier. S being a proof structure, it's frontier is made only of conclusions. So p_1 (resp. p_2) is connected to a conclusion α (resp. β) by a switching path τ_1 (resp. τ_2).

- If $\alpha = \beta$ then obviously p_1 and p_2 are connected in σS.
- If not, using the assumption, we can ensure there is a switching path τ from α to β. Then note that $\tau_1 \tau \tau_2$ is a switching path from p_1 to p_2. Meaning there is a switching μ such in μS the premisses p_1 and p_2 are connected.

In any case we show the connection of the premisses. This for any $⅋$ link, showing that S is saturated. □

Proposition 5.4. *Given any proof structure S that DR^- correct one of two cases occurs;*

- *(Connected). Every switching of S results in a connected graph. And so S is DR correct.*
- *(Conclusion Disconnection). There exists two conclusions α and β such that for any switching they are not connected.*

Proof. Here we differ from the argument of Bechet in [BEC98] which is really hard to get a sense of, also we seek to use to the same idea; that S cannot be an infinite graph. We will assume that S does not have the property of conclusion disconnection, note that given the last proposition 5.3 this means that, S is saturated. And we will want to show that S is connected after any switch.

Let us point out one thing, due to S saturation, for any $⅋$ node u of S and its two premisses α_1, α_2, there exists a switching path from α_1 to α_2 that for commodity we will denote $path_{⅋}(\alpha_1, \alpha_2)$.

Now let us show that disconnection can not happen, consider any switch σ of S. Take two components v_1, v_2 of σS, by DR^- correctness v_1 is connected to γ_1 by a path τ_1 while v_2 is connected to γ_2 by a path τ_2 in σS.

- If $\gamma_1 = \gamma_2$ then obviously v_1 and v_2 are connected in σS.
- If not, using the assumption, we can ensure there is a switching path τ from γ_1 to γ_2.

Now note that τ may not be a path for the switching σ, this would mean both premisse (α_1, α_2) of $⅋$ link are used in τ and τ_1 (or τ_2). Now we can edit the path τ such that to pass from α_1 to α_2 we travel according to $path_{⅋}(\alpha_1, 2)$.

For each $⅋$ nodes with its two premisses involved we do this operation, recursively. This operation is bounded since otherwise we would have an infinite amount of $⅋$ nodes, and S would not be a proof structure.

This way we can ensure that there is a path from v_1 to v_2 in σS. And so we ensured that S is connected.

□

Theorem 5.1 (Modules Correctness criterion)**.** *Given a module we have the equivalency;*

$$\mathcal{M} \textit{ is } DR^- \textit{ correct} \Leftrightarrow \mathcal{M} \textit{ is a sub–module of a proof net.}$$

Proof. **(2→1).** Assume that $\mathcal{M}$ is a sub module of a proof net S. Then a switching of $\mathcal{M}$ is the restriction of a switching of S, since $V_{\mathcal{M}} \subset V_S$. Given a switching σ of $\mathcal{M}$ then there exists μ a switch of S such that $\mu \mid V_{\mathcal{M}} = \sigma$.

By assumption S is a proof net, so μS is acyclic and connected. Therefor it follows that since $\mu S \subset \sigma S$ is also acyclic. Connection is ensure since we have $\mu S \subset \sigma S \subset S$ with μS and S connected.

σS is acyclic so is $\sigma\mathcal{M}$ since the reverse would lead to a direct contradiction.

Assuming there is internal component in $\sigma\mathcal{M}$ would lead to σS being disconnected. Since σS corresponds to an extension of $\sigma\mathcal{M}$ on which we can only connect new nodes to the frontier.

(1⇒2). *By induction.* We operate by induction on the number of conclusion and premisses of the module. We are given a correct module $\mathcal{M}$ and seek to complete it in a proof net.

If $\mathcal{M}$ has no conclusion, it can be made of cut nodes or deadlocks. A deadlock is not DR^- correct since it has a switching cycle. A cut node can be completed in a proof net, of a cut on two axioms $\vdash A, A^\perp$.

Now assume $\mathcal{M}$ has $n+1$ one conclusions, and that for any module with n conclusion the implication holds. Assume $\mathcal{M}$ is DR^- correct, this equivalently mean that $\mathcal{M}^+$ is DR^- correct thanks to the proposition 5.2. Now calling the proposition 5.4, we can ensure that either $\mathcal{M}^+$ is DR correct, or $\mathcal{M}^+$ has two conclusions that are disconnected for any switching.

If $\mathcal{M}^+$ is DR correct then this means it is a proof–net, and so $\mathcal{M}$ is a sub–module of a proof net, that is $\mathcal{M}^+$.

Now if $\mathcal{M}^+$ has two conclusions α and β that are always disconnected for any switching. Consider the structure $\mathcal{M}_1^+$ which consists of $\mathcal{M}^+$ where α, β are connected by a $\otimes$ link. This structure has no cycles, since $\mathcal{M}^+$ is acyclic, and α, β are disconnected for any switch, the placement of this new tensor node cannot generate cycles. Also $\mathcal{M}_1^+$ has no internal component since $\mathcal{M}^+$ does not.

$\mathcal{M}_1^+$ is therefore a DR^- correct, with one less conclusion than $\mathcal{M}$, by induction therefore, we can conclude $\mathcal{M}_1^+$ is sub–module of a proof net. Since $\mathcal{M}$ is a sub–module of $\mathcal{M}_1^+$, we can ensure that $\mathcal{M}$ is the sub–module of a proof net. □

Theorem 5.2 (Completion)**.** *Given a cut–free module $\mathcal{M}$ with one conclusion.*

If $\mathcal{M}$ is DR^- correct, then there exists a cut–free module $\mathcal{M}^*$ with dual entering arrows, such that connecting $\mathcal{M}$ and $\mathcal{M}^*$ by axioms results in a proof net.

Proof. The key point is to show that, the one conclusion of $\mathcal{M}$ is connected to all others elements of the frontier. More precisely denoting $\alpha_1, ..., \alpha_n$ the entering arrows and α the conclusion of $\mathcal{M}$. There exists always a switching path from α_i to α, this is straightforward since $\mathcal{M}$ is a tree α_i is always connected by a descending path to α, and, a descending path is necessarily a switching path.

Then assume $\mathcal{M}$ is DR^- correct, calling the proposition 5.2 then equivalently, $\mathcal{M}^+$ is DR^- correct and so calling the proposition 5.4, either $\mathcal{M}^+$ is a proof–net[28] and so the completion module $\mathcal{M}^*$ corresponds only to the dual arrows of the premisses of $\mathcal{M}$.

[28] Since it is DR correct in that case.

Or $\mathcal{M}^+$ has two disconnected conclusions β_1, β_2. But the conclusion α of $\mathcal{M}$ is connected to each of its premisses, meaning in $\mathcal{M}^+$, α is connected to each conclusions. There connecting β_1, β_2 by a tensor node creates a cut free[29] modules $\mathcal{M}^*$ that is connected to $\mathcal{M}$, and results in $\mathcal{M}_1^+$ that has 1 less conclusion.

Calling the induction like in the last theorem's proof we will conclude. □

[29] In fact this module will be made only of tensors.

6 Behaviors and The $\parr$–closure

We will now follow and focus on the article [BEC98] of D. Bechet. The motivation behind the work of Bechet is to question the meaning of the correctness criterion for the dynamic of proof structures, ie. the cut elimination. Bechet shows that there exists no bigger class of proof structures, other than proof nets, that cannot normalize to a bad configuration (that will be define).

Definition 6.1 (Rivality, Interaction, Modular combination)**.** Given a proof structure S of conclusions $\alpha_1, ..., \alpha_n$ respectively labeled by $A_1, ..., A_n$.

- We call *rivality* of S a family of proof structure $(S_1, \beta_1), ..., (S_n, \beta_1)$ such that, each S_i is a proof structure while β_i a conclusion of S_i that is labeled by $A_i^b ot$.
- Then we call their *interaction* the proof structure, obtained by linking each arrow α_i of S to the arrow β_i of S_i.
- Given a proof structure S a *modular combination* of S correspond to an interaction with a rivality made of proof–nets.

Definition 6.2 (Substitution function, Instantiation)**.** Given a *substitution function* defined on the MLL formulaes, $\phi : F_{MLL} \to F_{MLL}$, such that for any formula A, $\phi(A^\perp) = \phi(A)^\perp$.

Then for any proof structure S we define it's *instantiation* $\phi(S)$ as the same Quiver of which we only change the labeling. Such that each arrows α outgoing from an axiom labeled by A in S is labeled by $\phi(A)$ is S.

The label of the other types of arrows is then constructed by the 'labeling rules' of the MLL links[30].

Interaction, well and badly behaved structures. Since we want to study correctness under the scope of interaction (and so cut elimination), we state 3 interaction and transformations of proof structures (that preserve correctness).

We want to exhibit a class a proof structures that are well behaved with cut elimination, these structures should be stable by the 3 following transformation. Alternatively said, the class of well behaved structures is a class of proof structures, that is closed under the following transformation.

- (Cut elimination) Well behaved structures are stable by cut-elimination.
- (Modularity) The modular combination of a well behaved proof structure, is still well behaved.
- (Instantiation) Any instantiation of a well behaved object, remains well behaved.

We stated 3 transformations to capture the notion of well behaved structure. But, we will define the class negatively, ie. we will define structure that are not well behaved. Then well behaved ones will simply corresponds to structure that do not have a bad behavior.

[30] In fact the labeling of a MLL proof structures, can be given only by a labeling of its node, and a labeling of the axiom–outgoing arrows.

Disruptive proof structures. A structure that behaves badly is first merely any structure that is not stable by the cut elimination, modularity, or instantiation. Any structure that by one of these transformation becomes *badly behaved* (or we might use the word *disruptive*), will be, just as well, badly behaved.

Therefore we need to define the basic class of disruptive proof structures, which Béchet chooses to call *basically wrong proof structure* in the original article [BEC98].

Definition 6.3 (Basically wrong proof structure)**.** A proof structure is said to be *basically wrong*, if and only if, it is either a disconnected graph or a deadlock.

Then, Bechet introduces a terminology corresponding to each transformations that a proof structure can fail.

- Then we call *wrong* proof structures, any proof structure that reduces to a basically wrong proof structure by cut elimination (in n steps, n being possibly 0).
- A *basically bad* proof structure is a proof structure unstable by modularity. Meaning that we can make it interact (with cuts) with some proof nets, and it reduce by cut elimination to a wrong proof structure.
- A *bad* (or disruptive) proof structure is a proof structure not logically stable by instantiation. Meaning there exists a substitution of the atomic formulas that makes it it basically bad.

Note that we have inclusion of the 4 classes of disruptive proof structures. In fact one can note that failing cut elimination, ensures failing modularity (interact with axioms). And failing modularity ensures failing instantiation (consider the identity instantiation).

The point of the article of Bechet [BEC98], is then to show that the notion of incorrect proof structure and of disruptive proof structure are the same, for a structure without cuts.

Claim. *The disruptive proof structures (without cuts) are exactly the proof structures that are not proof nets.*

Correctness ensures good behavior. One of the direction of this equivalency is not too hard to show, it is based on the fact that correctness is preserved by cutelimination, by interaction, and by instantiation. This way we can show that a correct proof structure, is a well behaved one.

Theorem 6.1. *Given a proof structure S, if S is correct then necessarily it is not disruptive (ie. it is well behaved).*

Proof. Let's assume that S is correct, then indeed cannot be a deadlock or disconnected, it is therefore necessarily not basically wrong.

Then if S is correct any reduction (by cut elimination) of S is correct thanks to cut elimination preserving correctness.

If S interacts with some proof nets $P_1, ..., P_n$. We obtain again a correct proof structure (since the cuts are used in a *splitting* way), therefore by preservation of correctness again, it will not reduce to basically wrong proof structure.

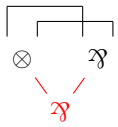


Figure 18: A ⅋ closure

Finally, instantiation preserves correctness, and $\phi(S)$ being correct all the previous arguments hold, meaning it cannot reduce to a basically wrong proof structure.

Therefore we conclude that S is not a bad proof structure. □

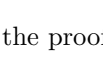
⅋-Closure and interactive behavior. A commodity in the study of the behaviors of proof structure, is that one can merely study the behavior of the *⅋-closure* of a proof structure S, ie. the structure obtained after linking each of the conclusion of S by a ⅋ node (in any given order), in order to conclude on the behavior of S. In fact we can show that S and its ⅋ closure have the same behavior.

Lemma 6.1. *Given a proof structure S and it's conclusions $A_1, ..., A_n$ (for any enumeration). The following are equivalent;*

1. *S is a bad proof structure*
2. *The structure obtained from S in which A_1 and A_2 have been linked by a ⅋ node is bad.*

Proof. **(1⇒2).** Let's assume that S is bad and let's consider T the structure obtained after doing a ⅋ link on two conclusions of S, let's say these conclusions are A_1 and A_2. Let's show that T is bad.

- (Base, deadlock) If S is a deadlock. Then S has no conclusions, so this case is excluded since T cannot exist.
- (Base, disconnection) If S is disconnected. Let us show that T fails modularity again. Indeed if we are given any environment $A_1^\perp, \Gamma_1, ..., A_n^\perp, \Gamma_n$ and their associated proofnets $P_1, ..., P_n$.

  Consider then $P =$  the proof net obtained from P_1 and P_2 by linking $A_1^\perp$ and $A_2^\perp$ by a $\otimes$ node. It is clear then that the interaction with a cut of T and P on $A_1 ⅋ A_2$ will reduce to a disconnected proof structure, ie. a basically wrong structure.
- (cut elimination)If S is simply wrong, ie. is bad because it fails cut elimination. Meaning $S \rightsquigarrow S'$ such that S' is basically bad. Then $T \rightsquigarrow T'$ where T' corresponds to S' with A_1 and A_2 linked by ⅋. But S' is disconnected so by induction, T' which is obtained by adding a ⅋ link is bad

aswell, since adding a $\parr$ node to a structure with a disconnected switching, still makes a structure with a disconnected switching.

- (Modularity) If S is bad since it fails modularity, meaning $\underbrace{S \quad P_1 \quad \cdots \quad P_n}$ $\rightsquigarrow$ W where W is wrong. Then indeed, T will also fail modularity, considering $\underbrace{T \quad P_1 \otimes P_2 \cdots \quad P_n}$, that will in one step reduce to $\underbrace{S \quad P_1 \quad \cdots \quad P_n}$, and so to W a wrong proof structure.
- (Instantiation) If S is bad because of instantiation, then indeed so is T since adding a $\parr$ node only affects output and not inputs.

(2⇒1). In this case we assume that T is bad and we want to show that S is bad.

- (base) If T is disconnected, indeed removing a $\parr$ node will conserve this. At the same time T cannot be bad for the reason of being a deadlock, since it contains a $\parr$ node, T cannot be a deadlock (although it could contain one, and therefore be a disconnected graph).
- (Cut elimination) If T is bad because it fails cut elimination, ie. there exists T' such that $T \rightsquigarrow T'$ and T' is a disconnected structure. Then indeed $S \rightsquigarrow S'$ where S' is obtained from T' removing the appropriate $\parr$ link. T' being disconnected so is S'.
- (Modularity) If T is bad because it fails modularity, it means there exists $Q_0, ..., Q_k$ proof nets such that $\underbrace{T \quad Q_0 \quad \cdots \quad Q_n}$ reduces to a basically wrong structure. We adapt the notation such that the conclusions of T are $B_0 = A_1 \parr A_2$ and $B_i = A_{i+2}$ for any index $i \neq 0$.

  Then the point is to know about Q_0, and especially about its output arrow labeled by $A_1^\perp \otimes A_2^\perp$ that is involved in the cut link. There can be two possibilities, either this arrows is the result of a $\otimes$ node, or it is the output of an axiom node.

  If the output arrow is linked to an axiom node, then after one reduction step the $\parr$ node is not involved in the reduction. If the output arrow is linked to $\otimes$, then we can conclude by confluence since $\underbrace{T \quad Q_0 \quad \cdots \quad Q_k}$ will reduce in one step to $\underbrace{S \quad P_1 \quad \cdots \quad P_n}$
- (Instantiation) If T is bad because it fails instantiation, ie. there exists an instantiation ϕ such that $\phi(T)$ fails modularity, then indeed by induction, so does $\phi(S)$. Meaning S fails instantiation.

□

Proposition 6.1. *Given a proof structure S, and S' one of its $\parr$ closure. The following statement carry the same consequences;*

1. *S is a disruptive proof structure.*
2. *S' is a disruptive structure.*

Proof. It is straightforward application of the lemma by induction on the number of conclusions of the given proof structure. □

Why cut free proof structures. At this point our goal is to show that an incorrect proof structure without cuts corresponds exactly to a bad proof structure. It makes sense to study proof structure without cuts, since every proof structure is normalising, and also since cut elimination preserves correctness. If still after cut elimination the structure is incorrect we show that it is the same as having a bad interaction with proof nets. The pathological case is the one of incorrect proof structures that are not bad ie. that reduce to a correct proof structure.

There is another justification to the study of cut–free proof structures. The geometry of the tensor and of the cut are in fact the same[31], therefore given a proof structure S[32] we can substitute each cut nodes of the structure by tensors, and obtain a proof structure without cuts. The switches of the obtained cut–free proof structures will geometrically be the same than the original structure S, therefore we could conclude that these two structures have *the same – geometrical – behavior*.

Two cases to study. For the equivalency, it remains to show that, an incorrect cut free proof structure S is a structure with bad interactive behavior. The idea is to consider two cases, the one where S contains a disconnected switching, and the one where it contains a switching cycle. In the first case, we show that there is a way to make S interact with an environment such that it reduces to – one of – its disconnected switching σS. In the second case we show that (an instantiation of) S can interact with a correct environment, that will then reduce to a structure containing a deadlock.

[31] They behave similarly after switching.
[32] And enventually a module, the principle remains.

7 Cut Elimination, duality and interactions

7.1 The heavyness of graph theory

Isomorphisms of quivers. Given two quiver what equality means with our intuition (two quiver with the same representation), corresponds formally to a notion of isomorphism, it is necessary to introduce such a notion to formally discuss of the equality of quivers. The point is that two quiver can have the same representation, but it is not said that the set of nodes or of arrow are really 'the same'.

We define an isomorphism between two quivers $G_1 = (V_1, Arr_1, src_1, tgt_1)$ and $G_2 = (V_2, Arr_2, src_2, tgt_2)$ as a pair of bijection $f : V_1 \to V_2$ and $g : Arr_1 \to Arr_2$. Such that $f(src_1(\alpha)) = src_2(g(\alpha))$ and $f(tgt_1(\alpha)) = tgt_2(g(\alpha))$ for each arrows $\alpha \in Arr_1$.

By an abuse we might say that two quivers are equal and denote $G_1 = G_2$ to mean that they are isomorphic.

Sub-module. Just like equality, the expression of being a sub module corresponds to the formal idea, of an injective morphism. We will say that $\mathcal{M}_0$ is a submodule of $\mathcal{M}$ and denote, $\mathcal{M}_0 \subset \mathcal{M}$ if and only if, there exists two function $(f : V_0 \to V_{\mathcal{M}}, g : Arr_{\mathcal{M}_0} \to Arr_{\mathcal{M}})$ such that, both are injective. And also such that $src(g(\alpha)) = f(src(\alpha))$ and $tgt(g(\alpha)) = f(tgt(\alpha))$ for each arrow $\alpha \in Arr_{\mathcal{M}0}$. And such that the labels of antecedent and images are the same.

Cut Elimination needs orientation. We first try to give a formal definition of cut elimination of proof structures, it corresponds to graph rewriting formally the definition can get heavy. Note that we equip proof structures – and modules –, of a function *or* called *orientation* such that *or* is of domain $V_{\mathcal{M}} \setminus V_{\mathcal{M}}^{(Ax)}$ ie. the nodes not labeled by an axiom. And of codomain $Arr^2_{\mathcal{M}}$.

The idea is that given a vertices v in the domain, $or(v)$ corresponds to a pair of arrow (α_1, α_2) such that $\alpha_1 \neq \alpha_2$ and both arrows target v, basically it orders the inputs of the vertice v.

Types of cuts. Given a vertice v labeled by a cut in some module $\mathcal{M}$, we say that v is a *degenerated cut* if given $or(v) = (\alpha_1, \alpha_2)$ the two inputs of v, have the same source, $src(\alpha_1) = src(\alpha_2)$.

We also distinguish to types of nodes in order to define cut elimination;

- (Normal) We say that v is a *normal cut* if given $or(v) = (\alpha_1, \alpha_2)$ one of the source of α_1 or of α_2 is labeled by an axiom.
- (Multiplicative) We say that v is an *multiplicative cut* if it isn't normal.

The first thing to point out is that we will not define cut elimination on degenerated cuts. The second thing to note is that a multiplicative cut necessarily corresponds to a cut of a $\parr$ against a tensor $\otimes$.

Formal Cut Elimination. Given a vertice v that is a non degenerated cut in some module $\mathcal{M} = (V_{\mathcal{M}}, Arr_{\mathcal{M}}, src_{\mathcal{M}}, tgt_{\mathcal{M}})$. We define the reducted module $rd(\mathcal{M}, v)$ the module obtained after eliminating the cut node v distinguishing two cases.

If v is a normal cut, assuming $src(\alpha_1)$ is labeled by an axiom let's denote by (α_1, β_1) the two output of the axiom node, we define $rd(\mathcal{M}, v)$ as;

- $V_{rd(\mathcal{M},v)} = V \setminus \{v, src(\alpha_1)\}$, we remove the cut node and the axiom involved.
- $Arr_{rd(\mathcal{M},v)} = Arr_{\mathcal{M}} \setminus \{\alpha_1, \beta_1\}$, ie. we remove the two output of the axiom node involved in the cut.
- $src_{rd(\mathcal{M},v)} = src_{\mathcal{M}}$ the source is unchanged.
- $tgt_{rd(\mathcal{M},v)}(\alpha_2) = tgt_{\mathcal{M}}(\beta_1)$ and for any other arrow the target is unchanged.

If v is a multiplicative cut, assuming $or(src(\alpha_1)) = (\beta_1, \beta_2)$ and $or(src(\alpha_2)) = (\gamma_1, \gamma_2)$. We define $rd(\mathcal{M}, v)$ as

- $V_{rd(\mathcal{M},v)} = V \setminus \{v, src(\alpha_1), src(\alpha_2)\} \cup c_1, c_2$, we remove the cut node and the nodes involved. And we add two new cut nodes.
- $Arr_{rd(\mathcal{M},v)} = Arr_{\mathcal{M}} \setminus \{\alpha_1, \alpha_2\}$, ie. we remove the two arrows involved in the cut.
- $src_{rd(\mathcal{M},v)} = src_{\mathcal{M}}$ the source is unchanged.
- $tgt_{rd(\mathcal{M},v)}(\beta_1) = tgt_{\mathcal{M}}(\gamma_1) = c_1$ and $tgt_{rd(\mathcal{M},v)}(\beta_2) = tgt_{\mathcal{M}}(\gamma_2) = c_2$. For any other arrow the target is unchanged.

Proposition 7.1 (Dual confrontation normalize into an indentity cut module)**.** *Given a $\otimes, \parr$ tree $\mathcal{T}$, the confrontation $\mathcal{T} \llcorner\lrcorner \mathcal{T}^{\perp}$ reduces to a module of cuts (following the identity).*

Proof. By Induction. We reason by induction on the number of vertices of the tree. If $\mathcal{T}$ has one vertices, assuming it is labeled by $\otimes$ (without loss of generality) then it's dual consist of on node label by $\parr$. Let's denote the node of $\mathcal{T}$ by u and the one of the dual by v, and c the cut node of $\mathcal{T} \llcorner\lrcorner \mathcal{T}^{\perp}$, denote $or(u) = (\alpha_1, \alpha_2)$ and $or(v) = (\alpha_1^{\perp}, \alpha_2^{\perp})$.

Then the cut node c is a multiplicative cut node, and it reduces $\mathcal{T} \llcorner\lrcorner \mathcal{T}^{\perp}$ into a module $\mathcal{M}$, such that by definition, there remain only two cut nodes c_1 and c_2 such that c_1 connects $(\alpha_1, \alpha_1^{\perp})$, and c_2 connects $(\alpha_2, \alpha_2^{\perp})$.

So $\mathcal{M}$ consist of a cut module defined by the application $\alpha_1 \mapsto \alpha_1^{\perp}$ and $\alpha_2 \mapsto \alpha_2^{\perp}$.

Now assume the proposition is true for trees with n vertices, and take a tree of $n+1$ vertices, then two cases are possible, either the root is labeled by $\otimes$ or by a $\parr$. We can assume the root denoted r is labeled by a tensor without loss of generality, $\mathcal{T}$ is then the connection of two subtrees $\mathcal{T}_1$ and $\mathcal{T}_2$. Denote c the cut node of $\mathcal{T} \llcorner\lrcorner \mathcal{T}^{\perp}$, it links the conclusions γ and $\gamma^{\perp}$ of each tree.

Being a multiplicative cut node the elimination of c will result in two cut nodes connecting respectively $(\gamma_1, \gamma_1^{\perp})$ and $(\gamma_2, \gamma_2^{\perp})$, where γ_1 is the conclusion of $\mathcal{T}_1$ and $\gamma_1^{\perp}$ the conclusion of $\mathcal{T}_1^{\perp}$ (and the same for γ_2 and $\gamma_2^{\perp}$). It therefore

results in two modules $\mathcal{T}_1 \llcorner\!\lrcorner \mathcal{T}_1^{\perp}$ and $\mathcal{T}_2 \llcorner\!\lrcorner \mathcal{T}_2^{\perp}$ that we know reduce to cut modules according to the map $\alpha \mapsto \alpha^{\perp}$. □

7.2 Modules cut-elimination as rewriting, preliminary ideas from ludics

An inadapted Formalism, and towards ludics. We can proove properties of the cut elimination using the formal framework of graph theory, but, we can rapidly notice that it is not the most adapted choice to study cut elimination, it simply gets too heavy, and maybe worse, the difficulty resides in expressing formally how the set of vertices, the arrows the target and the source are edited. The difficulty is not *really* about what happens in the process of cut elimination but more in *how* we express it.

This trouble, might be one the reason why Girard introduced ludics in "Locus Solum" [Gir02] in 2002, a field that is still young. We will not get exactly into ludics, as it would need an introduction on its own, but we refer to the paper of Curien entitled "Introduction to Ludics part II", see [Cur05]. It is interesting to note that like Bechet studied $⅋, \otimes$ trees, Curien suggests here to study modules that are cut and axiom free, as forests, of which each tree is a representation of a formula.

⅋,⊗-Trees. We will call ⅋,⊗-Trees any module $\mathcal{T}$ that is labeled only by ⅋ and ⊗. And such that this module has one conclusion. These trees also corresponds to bodies of proof structures without cut and that have one conclusion. We will focus on studying these trees to then conclude on proof structures behavior. Since a proof structure behavior is equivalent to the behavior of its ⅋-closure, it makes sense to study these trees (following the idea of Bechet). Following the idea of Curien it makes sense on its own to isolate trees since they will be the component of the forest.

Another reason to study them is that, the structure of tree enjoys a good property of induction, indeed we can always look at the root of a tree, and see how the remaining subtrees behave. Induction is possible on proof structures, but would be a lot more subtle and hard to manage well (for exemple removing a terminal node does not conserve the fact that the structure has one conclusion, which is what some of our arguments are based on).

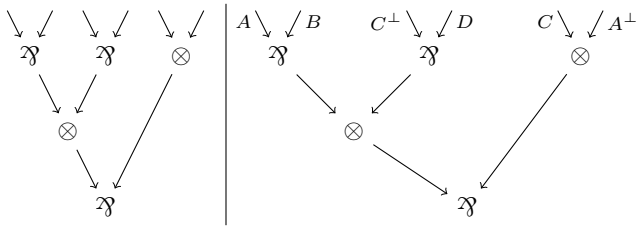


Figure 19: a $\otimes, \parr$-tree, typed and untyped.

Formulas as modules. The point of this observation is to show that any formula A can be represented by eventually many modules made of one conclusion arrow labeled by A. A module made of one conclusion A, without axioms nor cuts, will be called a representation of the formula A.

Let us introduce the straightforward grammar, of expressions denoting a representation module. Where A is any formula of **MLL** (eventually not atomic).

$$E, F = \overrightarrow{A} \mid E \otimes F \mid E \parr F \mid E^{\perp}$$

To each of these expression E we can in a straightforward way associate a representation $[\![E]\!]$ of a formula.

To $\overrightarrow{A}$ we associate the module corresponding of one arrow labeled by the formula A. Then the expression $E \otimes F$ corresponds to, the module made of one $\otimes$ node, connecting the two modules $[\![E]\!]$ and $[\![F]\!]$ respectively by A and B. In a similar way we define the module corresponding to $\overrightarrow{A} \parr \overrightarrow{B}$. For the module corresponding to the *negation* of an expression $E^{\perp}$, it simply corresponds to the dual graph of $[\![E]\!]$.

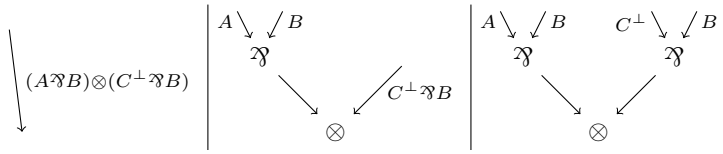


Figure 20: Possible representations of the formula $(A \parr B) \otimes (C^{\perp} \parr B)$.

Note that to a formula corresponds many expressions, and so just as many trees. We can wonder on the number of representation of a formula. We claim that a formula of **MLL** as just as many representation that it has connectors $\parr$ or $\otimes$, plus one. Indeed an atomic formula has 1 representation, while by induction the number of representations is incremented by one, just as the number of operator.

Also Note note that module representations of formulaes of **MLL**, corresponds to $\otimes, \parr$ trees. And even further; any $\otimes, \parr$ tree $\mathcal{T}$ corresponds to the

representation of a formula A, where A is the conclusion of $\mathcal{T}$.

Proof structure bodies are formulas representation. We note that a module $\mathcal{M}$ made of only $⅋$ and $\otimes$ node, containing no axiom and no cut node, made of *one conclusion* is necessarily the representation of some formula. So note that, if S is a structure made of one conclusion A, then its body $\mathcal{B}_S$ is a $\otimes, ⅋$-tree that corresponds to a representation of A.

Axiom free and cut free modules as forests. We want to point out an idea that can be found in the paper of Curien [Cur05], where he gives an introduction to ludics. The idea is to notice that a module $\mathcal{M}$ without axioms nor cuts of conclusions $A_1, ..., A_n$, corresponds to a forest made of trees $\mathcal{T}_i$ each one being one of the (possibly many) representations of the conclusion formula A_i. Our next step is therefor to give a better description of formulaes representation.

Occurences. As pointed out before, the representation of a formula is not unique, to be fully describing this types of modules, we can introduce the notion of *occurence*. An occurence is a word u over the set $\{1, 2\}$, then given a formula A we call the subformula at occurence u of A and denote by A/u the subformula obtained by induction by the following definition;

- $A/\epsilon = A$
- $(A \square B)/1u = A/u$ where $\square = \otimes$ or $\square = ⅋$.
- $(A \square B)/2u = B/u$ where $\square = \otimes$ or $\square = ⅋$.
- If $X \in \mathcal{V}$, $X/1 = X/2 = \bot$ the proper[33] subformula of an atomic formula are undefined.

We say that two occurence u and v are *comparable* if, u is a subsequence of v or v is a subsequence of u. Two occurence that are not comparable are said to be *disjointed*. Given a formula A we say that a set U of occurences is *total*, if and only if,

- For any occurence $u \in U$, A/u is defined.
- $subformula(A) = \bigcup_{u \in U} subformula(A/u)$.

We then call *partition* of a formula A (in reference to set theory), a set of disjoint occurences that is total w.r.t. A.

From trees to partitioned formula. We pretend that, a formula A and one of its partition U corresponds to a representation tree. But also the reverse that to a tree always corresponds a partition of the formula.

Given U a set of occurence, and given v any occurence we will denote by $U^{(v)}$ the set of occurence of u such that v is a subsequence of u. Given a formula A and U a partition of A, we will denote by $[U : A]$ the $⅋, \otimes$–tree obtained by induction where;

- $[\epsilon : A]$ corresponds to $\overrightarrow{A}$ the arrow labeled by A
- If $F = A \square B$, $[U : F] = [U^{(1)} : A] \square [U^{(2)} : B]$ with $\square = \otimes$ or $\square = ⅋$.

[33] We mean that a subformula of some formula A is a proper subformula if it isn't A.

Locative Structure. To involve cuts (but also axioms) there is a need to involve the notion of location, as Curien says in [Cur05] an idea of ludics is to merely keep the *locative structure* of the proof structures. We therefor need to introduce the notion of *relative adresses* (the terms comes from ludics). We call *positioning* of a forest F a injective function $pos : F \to \mathbb{N}^*$ where $\mathbb{N}^*$ is intended to be the set of finite sequences of integer. We can always ensure its existence since we assume the forest to be finite.

Instead of considering the positioning as an outside function we will directly mention it in trees, we introduce the triplet of the form $[\xi, U : A]$ where ξ is an adress, and U a partition of the formula A. And we will denote forests as

$$[\xi_1, U_1 : A_1], ..., [\xi_n, U_n : A_n]$$

Where each adresses $\xi_1, ..., \xi_n$ are distinct.

There is no canonical positioning for a given proof structure, it is also interesting to note, that as pointed out Bechet, we can use the $\parr$ closure to make a forest into a tree, then the positioning corresponds to how we choose to place the $\parr$ nodes in the closure.

Configurations and adresses. The configuration defined for a module as dijoint pairs of input arrows, can in this new framework be expressed. Given a forest $[U_1 : A_1], ..., [U_n : A_n]$ a configuration $\mathcal{C}$ corresponds to dijoint pairs of $\bigcup_{1 \leq i \leq n} U_i$. Although there might be ambiguity if one we do not involve relative adresses.

Therefor given a forest with relative adresses, $[\xi_1, U_1 : A_1], ..., [\xi_n, U_n : A_n]$, a (non ambigious) configuration correspond to disjoint pairs of $\bigcup_{1 \leq i \leq n} \xi_i . \bigcup_{1 \leq i \leq n} U_i$.

To invole cuts the process is similar, we define for such a forest a set D called *dynamic* of disjoint pairs of relative adresses, ie. of $\{\xi_1, ..., \xi_n\}$. The idea is that a cut node is relating each conclusion of the trees located at ξ_i and ξ_j if $\{\xi_i, \xi_j\} \in D$.

Then we can fully describe modules as triplets $[D]F[\mathcal{C}]$, where F is a forest, D a dynamic (pairs of relative adresses) of F and $\mathcal{C}$ a configuration of F. For the sake of readability we might use the virgula ',' to denote the union, and denote such triplets $[D]F[\mathcal{C}]$ as seen in curien paper [Cur05], by;

$$\begin{array}{c} \mathcal{C} \\ \hline F \\ \hline D \end{array}$$

Cut elimination as rewriting, the redexes. We can then define in this new framework the cut elimination process of MLL. First we want to introduce some notations, the product $\times$ will denote the opposition or confrontation of two trees,

$$[\xi, U : A] \times [\nu, V : A^\perp]$$

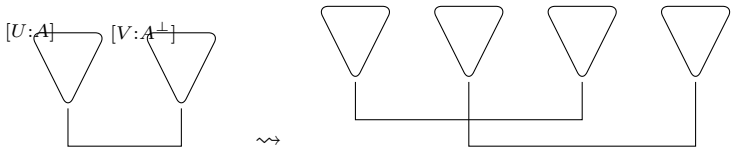


Figure 21: Illustration of the multiplicative redex

will denote

$$[\{\{\xi, \nu\}\}]\{[\xi, U : A], [\nu, V : A^{\perp}]\}.$$

While the addition notation will denote the union, ie. iven two modules, we will denote by $[D]F[C]+[D']F'[C']$ their union defined as $[D \cup D']F \cup F'[C \cup C']$.

For the sake of readability, given a tree $\mathcal{T} = [U : A]$ we denote by $\xi\mathcal{T}$ the triplet $[\xi, U : A]$. Then given an occurency $u \in U$ we denote by $\xi\mathcal{T}(u)$ the triplet $[\xi u, U^{(u)} : A/u]$. Furthermore using this notation we can explicit the dual tree as $\mathcal{T}^{\perp} = [U : A^{\perp}]$.

We will let the cut allowed only on trees that have dual labels. So we assume we are given two trees $\mathcal{T}_1 = [U : A]$ and $\mathcal{T}_2 = [V : A^{\perp}]$, where U, V are two partitions of A[34].

The first scenario of *multiplicative cut* corresponds to being given two partitions U and V that are both non empty, and the reduction is then,

$$\xi\mathcal{T}_1 \times \nu\mathcal{T}_2 \quad \rightsquigarrow_m \quad \xi\mathcal{T}_1(1) \times \nu\mathcal{T}_2(1) + \xi\mathcal{T}_1(2) \times \nu\mathcal{T}_2(2)$$

The scenario of *normal cut* is a little harder to describe, one of the key idea for this reduction is that an adress ν is at both time a relative adress and an occurency to do so, it means the tree in location ν correspond to a partition ϵ. Precisely it correspond to the following redex;

$$\dfrac{\dfrac{\{\{\nu, \mu u\}\}}{[\xi, U : A][\nu, \epsilon : A^{\perp}][\mu, V, B]}}{\{\{\xi, \nu\}\}} \quad \rightsquigarrow_{\epsilon} \quad [\xi', V \setminus \{u\} \cup u.U : B]$$

where ξ' can be any of the three adresses ξ, ν, μ.

Cut elimination as rewriting, the closure. The relation of cut elimination denoted $\rightsquigarrow$, corresponds to the transitive closure of the two previous given redex. We also call for cut elimination to be stable with union (this is the idea of locality of the cut elimination). Namely if $[D]F[\mathcal{C}]$ reduces to $[D']F'[\mathcal{C}']$, then given a second module $[D_0]F_0[\mathcal{C}_0]$ such that $D_0 \cap D = F_0 \cap F = \mathcal{C} \cap \mathcal{C}_0 = \emptyset$, then $[D]F[\mathcal{C}] + [D_0]F_0[\mathcal{C}_0]$ reduces to $[D']F'[\mathcal{C}'][D_0]F_0[\mathcal{C}_0]$.

But we also call for cut elimination to be compatible with re attributions of relative adresses. Formally we introduce an equivalency, and we say two

[34] and so equivalently of the dual formula $A^{\perp}$.

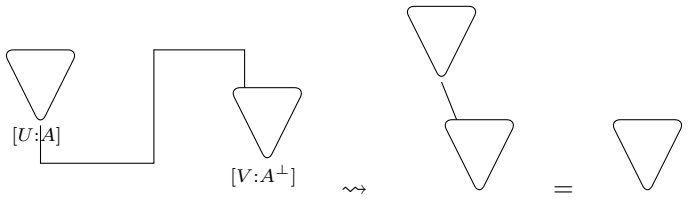


Figure 22: Illustration of the normal redex

modules $[D]F[\mathcal{C}]$ and $[D']F'[\mathcal{C}']$ are equivalent, if there exists a positioning pos_0 of the forest F such that $F = F'$, such that $\mathcal{C}' = \{\{pos_0 pos^{-1}\xi, pos_0 pos^{-1}\nu\} \mid \{\xi, \nu\} \in \mathcal{C}\}$, and the same for D.

Then cut elimination is compatible with equivalence in the sense that $\mathcal{M} \rightsquigarrow \mathcal{M}'$ and $\mathcal{M}' \equiv \mathcal{M}_0$ ensures $\mathcal{M} \rightsquigarrow \mathcal{M}_0$

Proposition 7.2 (Dual Confrontation)**.** *Given a cut free and axiom free MLL tree* $[A : U]$.

The dual confrontation, meaning the module $[\xi, U : A] \times [\nu, U : A^\perp]$ *reduces to;*

$$\sum_{u \in U} [\xi u, \epsilon : A/u] \times [\nu u, \epsilon : A^\perp/u]$$

Proof. By Induction. We reason by induction on the cardinal of U. If $U = \{\epsilon\}$ the proposition is immediatly true. Otherwise note that by definition of cut elimination (and also since U is nonempty) we have the reduction;

$$\xi\mathcal{T} \times \nu\mathcal{T}^\perp \quad \rightsquigarrow \quad \xi\mathcal{T}(1) \times \nu\mathcal{T}^\perp(1) + \xi\mathcal{T}(2) \times \nu\mathcal{T}^\perp(2)$$

Where $\mathcal{T} = [U : A]$. These two submodules obtained after reduction correpond to $[\xi j, U^{(j)} : A/j] \times [\nu j, U^{(j)} : A/j]$ (where j ranges in $\{1, 2\}$) so these – by induction since $U^(j)$ has less element than U – respectively reduce to

$$\sum_{u \in U^{(1)}} [\xi u, \epsilon : A/u] \times [\nu u : \epsilon : A^\perp/u] \quad \text{and} \quad \sum_{u \in U^{(2)}} [\xi u, \epsilon : A/u] \times [\nu u : \epsilon : A^\perp/u]$$

And so, since cut elimination is compatible the initial module reduces to their sum. Finally since $U = U^{(1)} \cup U^{(2)}$, their sum is

$$\sum_{u \in U} [\xi u, \epsilon : A/u] \times [\nu u : \epsilon : A^\perp/u].$$

So we can conclude. □

8 Case for disconnected switching

In this section, we want to treat one the cases of the implication, given S an incorrect cut–free proof structure, S has a bad behavior. We treat here the case in which S has a disconnected switching.

Dual graph. Given any graph G labeled by $\{\otimes, \parr, \mathbf{cut}, \mathbf{ax}\}$ we will call *dual graph* of G, and we will denote $G^\perp$, the graph identical to G such that each $\parr$ label becomes a $\otimes$ label, and each $\otimes$ label becomes a $\parr$ label. And such that each arrow labeled by a formula A in G, is labeled by $A^\perp$ in $G^\perp$.

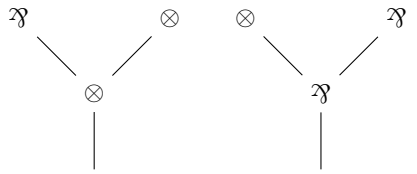


Figure 23: A $\otimes, \parr$ tree and its dual

Definition 8.1 (Body of a proof structure)**.** Given a structure S we will denote by $\mathcal{B}_S$ the module obtained after removing each axiome node from S, and setting the source of the output of axioms as undefined. Formally, defining and modules – and so proof structures – as quivers;

- $(V_S, Arr_S, tgt_S) = (V_{\mathcal{B}_S}, Arr_{\mathcal{B}_S}, tgt_{\mathcal{B}_S})$
- Given any arrow $\alpha \in Arr_S$, if $src_S(\alpha) \overset{\ell}{=} \mathbf{ax}$ then, $src_{\mathcal{B}S}(\alpha) = src_S(\alpha)$. Otherwise $src_{\mathcal{B}S}(\alpha) = \perp$ is undefined.

We want to stress that, if S has one conclusion and no cuts, its body $\mathcal{B}_S$ corresponds to a tree labeled by $\parr$ and $\otimes$. We will use this fact many times.

Modules interactions. We introduce two ways of how module cant interact. Consider that a modules has a set of input and output arrows, so interaction will be based on these arrows. Note that a module without any output nor input cannot interact [35].

We can therefor think of interactions of two modules $\mathcal{M}_1$ and $\mathcal{M}_2$, as partial functions that sends output or input arrows of $\mathcal{M}_1$ to input or ouput arrows of $\mathcal{M}_2$. We can this way define 3 basic types of interaction : An output-output interaction is called a *confrontation* or a *cut interaction* (or if there is no ambiguity, simply interaction). An input-input interaction is called a *linking* or an *axiom interaction*. Finally an input output interaction is called a *composition*.

[35] the main example being the deadlock. This is one the main reason we reject this proof structure as being well behaved, since it cannot even interact.

Confontation and linking interaction are said to be *valid* or *well-typed*[36] if and only if it sends an arrow labeled by A to an arrow labeled by $A^\perp$. Whereas a composition is valid if it sends arrow labeled by A to arrow labeled by A.

Interaction denotations. Given two modules $\mathcal{M}_1$ and $\mathcal{M}_2$ and ℓ one of their link. We will denote by $\mathcal{M}_1 \overset{\ell}{\urcorner} \mathcal{M}_2$, the module obtained after linking each input arrow i of $\mathcal{M}_1$ to $\ell(i)$ of $\mathcal{M}_2$ by an axiom node.

In this case we can link the two modules by adding an axiom node. We will denote the 1-link $\mathcal{M}_1 \overset{i}{\underset{j}{\urcorner}} \mathcal{M}_2$ the module obtained after linking the input arrow i of $\mathcal{M}_1$ to a compatible input j of $\mathcal{M}$. Adding an axiom node that becomes the source of the two arrows i and j.

We might also denote $\mathcal{M}_1 \overset{A}{\underset{A^\perp}{\urcorner}} \mathcal{M}_2$ to mean that we link to arrows of type A and $A^\perp$, but this notation is not explicit, since we do not give information on which input are affected (there could be several input of type A or $A^\perp$).

We will denote by $\mathcal{M}_1 \urcorner \mathcal{M}_2$ the (or we should say 'one of the') proof structure obtained after linking each input of $\mathcal{M}_1$ to inputs of $\mathcal{M}_2$ by the means of an axiom node, note that this could be done in several ways, not just one. Therefore also this notation isn't explicit, but we might use it if the distinction of how we pair inputs of $\mathcal{M}_1$ and $\mathcal{M}_2$ does not matter, or if there is no confusion to what we mean.

We define in a similar way the same notions for the output arrows and the cut node, the structures $\mathcal{M}_1 \overset{c}{\llcorner\lrcorner} \mathcal{M}_2$, $\mathcal{M}_1 \overset{o_1}{\underset{o_2}{\llcorner\lrcorner}} \mathcal{M}_2$ and $\mathcal{M}_1 \llcorner\lrcorner \mathcal{M}_2$, note that in the case of modules with only one conclusion there is no ambiguity in this last notation.

We do the same thing for composition denoting $\mathcal{M}_1 \overset{c}{\rightrightarrows} \mathcal{M}_2$ or $\mathcal{M}_2 \overset{c}{\leftleftarrows} \mathcal{M}_1$, where c is a composition interaction.

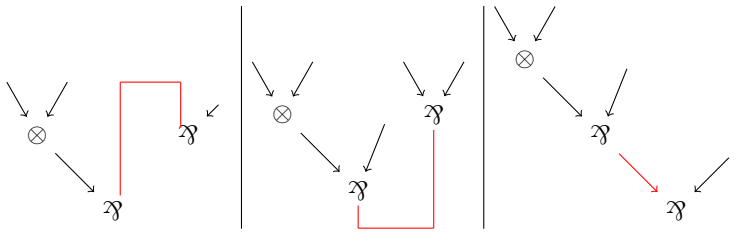


Figure 24: Modules interactions. A linking, a confrontation, and a composition.

Proposition 8.1 (Switching disconnection for ⅋,⊗-Trees)**.** *Given a tree $\mathcal{T}$ made of ⅋ and ⊗ node.*

1. $\mathcal{T}$ has a disconnected switching.

[36] We borrow this terminology from lambda-calculus, having in mind that formulaes corresponds to types, refering to the Curry-Howard correspondancy.

2. $\mathcal{T}$ has a $⅋$ node.
3. All switchings of $\mathcal{T}$ are disconnected.

Proof. **(1⇒2).** Indeed if $\mathcal{T}$ is made only of $\otimes$, then we can easily ensure that all switchings of $\mathcal{T}$ are connected. Formally we could do it by inclusion on the size of the tree, but the argument is straightforward since $\otimes$ connections are 'conserved' by switching.

So by contraposition, if $\mathcal{T}$ has a disconnected switching then, $\mathcal{T}$ contains at least one $⅋$ node.

(2⇒3). Now if $\mathcal{T}$ has a $⅋$ node, indeed no switching of the tree $\mathcal{T}$ can be connected. This can be ensured because trees contain no cycle, and so if a $⅋$ creates disconnection of two subtrees, then these two subtrees cannot 'reconnect' ie. be connected in $\mathcal{T}$, since it would mean $\mathcal{T}$ is not a tree.

(3⇒1). Since there always exists a switch, it follows. □

Arrow and Cut Modules. We will call *arrow module*, any module $\mathcal{M}$ made only of arrows. A module containing only cut nodes will be called a *cut module*. A cut module can always be seen as an arrow module on which we apply a *cut configuration*.

Canonical interaction with the Dual. Given a tree $\mathcal{T}$ we define a canonical interaction between $\mathcal{T}$ and $\mathcal{T}^\perp$. Denoting $i_1, ..., i_n$ the inputs of $\mathcal{T}$, and $i_1^\perp, ..., i_n^\perp$ the corresponding inputs in $\mathcal{T}^\perp$. The canonical interaction might also be called identity, and will be denoted id, it sends i_k to $i_k^\perp$. Note that this interaction is valid as linking but also as an opposition (identity for outputs).

Proposition 8.2 ($\otimes, ⅋$-Trees generate proof nets)**.** *Given a tree $\mathcal{T}$ made of $⅋$ and $\otimes$ node. And given σ a switching of $\mathcal{T}$.*

Then the structure $\mathcal{T}^\perp \overset{\ulcorner\urcorner}{_{id}} \sigma\mathcal{T}$ is a proof net, ie. a correct proof structure.

Proof. By induction. We will reason by induction on the trees size.

Base. Assume $\mathcal{T}$ is made of one node, it is then either a $⅋$ or a $\otimes$ node. Then the structure obtained corresponds to the proof of a $\otimes$ on two axioms, it is a correct structure.

Induction. By induction now, the tree $\mathcal{T}$ has a terminal node that is either $⅋$ or a $\otimes$ node, connecting two proper subtrees $\mathcal{T}_1$ and $\mathcal{T}_2$. Since the two proper subtrees have less nodes than $\mathcal{T}$ does, we can ensure by induction hypothesis that, $\mathcal{T}_1^\perp \ulcorner\urcorner \sigma\mathcal{T}_1$ and $\mathcal{T}_2^\perp \ulcorner\urcorner \sigma\mathcal{T}_2$ are proof nets.

Tensor-root. If the root of $\mathcal{T}$ is an $\otimes$ node. Then $\sigma\mathcal{T}$, corresponds to the connection of $\sigma\mathcal{T}_1$ and $\sigma\mathcal{T}_2$ by a $\otimes$ node. And $\mathcal{T}^\perp$ corresponds to the connection of $\mathcal{T}_1^\perp$ and $\mathcal{T}_2^\perp$ by a $⅋$ node.

Now let's show that given μ a switching of $\mathcal{T}^\perp \ulcorner\urcorner \sigma\mathcal{T}$ it will follow that $\mu(\mathcal{T}^\perp \ulcorner\urcorner \sigma\mathcal{T})$[37] does not contain cycles and is connected. Connection is

[37] In fact note that

ensured since an $\otimes$ node links $\mathcal{T}_1^\perp \upharpoonright \sigma\mathcal{T}_1$ and $\mathcal{T}_2^\perp \upharpoonright \sigma\mathcal{T}_2$ and that these two are knows to have only connected switchings. Acyclicity is ensure since the two substructure are proof nets and these two substructures are only linked by a $\parr$ and $\otimes$.

Parr-root. If the root node of $\mathcal{T}$ is a $\parr$ node, then $\mathcal{T}^\perp \upharpoonright \sigma\mathcal{T}$ corresponds to the nets $\mathcal{T}_i^\perp \upharpoonright \sigma(\mathcal{T}_i)$ connected by an $\otimes$ connecting the two conclusions of $\mathcal{T}_1^\perp$ and $\mathcal{T}_2^\perp$. And connecting by a $\parr$ the conclusions of $\sigma\mathcal{T}_1$ and $\sigma\mathcal{T}_2$. So indeed it is a proof net.

□

Proposition 8.3 (Cut elimination in the interaction of Dual Trees)**.** *Given a tree $\mathcal{T}$ made of $\otimes$ and $\parr$ node. $\mathcal{T} \llcorner\lrcorner \mathcal{T}^\perp$ reduces to a module being only cut modules, ie input arrows targeting a cut node.*

More precisely, it reduces to the modules made only of the input arrows $i_1, ..., i_n, i_1^\perp, ..., i_n^\perp$ that are linked by cut nodes, following the canonical interaction .

Proof. By Induction. We use induction on the size of the trees. Given any tree $\mathcal{T}$ made of one node. If so, then $\mathcal{T}$ is either made of a $\parr$ or a $\otimes$ node. $\mathcal{T} \llcorner\lrcorner \mathcal{T}^\perp$ will reduce in to cut modules.

Now consider $\mathcal{T}$ as connecting two trees $\mathcal{T}_1$ and $\mathcal{T}_2$. $\mathcal{T} \llcorner\lrcorner \mathcal{T}^\perp$ results, after one step of cut elimination, in the diconnected module made of $\mathcal{T}_1 \llcorner\lrcorner \mathcal{T}_1^\perp$ and $\mathcal{T}_2 \llcorner\lrcorner \mathcal{T}_2^\perp$, by induction they reduce into cut modules, by uniqueness of the normal form we can conclude. □

Proposition 8.4. *Given a proof structure S without cuts and made of one conclusion, and given σ any of its switchings. Then $\mathcal{B}_S^\perp \upharpoonright \sigma\mathcal{B}_S$ is a proof net.*

Proof. Remark that $\mathcal{B}_S$ is a $\otimes, \parr$ tree. We then simply call the proposition 8.2 to ensure that $\mathcal{B}_S^\perp \upharpoonright \sigma\mathcal{B}_S$ is a proof net. □

Proposition 8.5. *Given a proof structure S without cuts and made of one conclusion. σ a switching of S. The connecting structure $S \llcorner\lrcorner \mathcal{B}_S^\perp \upharpoonright \sigma\mathcal{B}_S$ reduces exactly to σS.*

Proof. Noticing that $\mathcal{B}_S$ is a $\parr, \otimes$-tree, thanks to the proposition 8.3, $S \llcorner\lrcorner \mathcal{B}_S^\perp$ reduce into the connection of cut and axiom nodes. Since the cut is local there remain the input arrow so that it is connected to $\sigma(\mathcal{B}_S)$. This reduce to $\sigma(S)$ simply by eliminating the succession of cut and axiom nodes. □

Finally we can ensure one the lemma's.

Lemma 8.1. *Given a proof structure S without cut, if there exists $\sigma(S)$ a disconnected switching of S.*

Then, S is a bad proof structure since it fails modularity.

Proof. Thanks to the proposition 8.4, $\mathcal{B}_S^{\perp} \parr \sigma\mathcal{B}_S$ is a proof net, ie. a correct environment. If S has good interation properties, then it should when interacting with a correct environment reduce to correct proof structure.

But refering to proposition 8.5, $S \lfloor\rfloor \mathcal{B}_S^{\perp} \parr \sigma\mathcal{B}_S$ reduces to the structure σS.

Since σS is a disconnected structure, it is in particular, a basically wrong proof structure. This shows that S is a bad proof structure, since it fails modularity, with the good environment that is $\mathcal{B}_S^{\perp} \parr \sigma\mathcal{B}_S$. □

9 Case for non-acyclic switching

9.1 Cyclicity, acyclicity and duality

In this subsection we explicit a transformation that we could call *Bechet's transformation*. It transforms a cut free module into another – also cut free – module. To explicit this transformation, we will introduce the notion of *configuration*. This transformation also correspond to a function that sends a module and one of its (minimal) switching cycle to a module that has no switching cycle. In this subsection we merely study the geometrical aspects of proof structure and their switching, we do not consider the well typedness of the axiom nodes (meaning that the conclusion of an axiom are dual).

Using the terminology that we will expose in this subsection. The bechet transformation takes a triplet $(\mathcal{M}, \mathcal{C}, T)$ such that;

- $\mathcal{M}$ is a module that is cut free and axiom free.
- $\mathcal{C}$ is a configuration of $\mathcal{M}$.
- T is a tracker of $\mathcal{C}$.

And it sends it to the triplet $(\mathcal{M}^\perp, T(\mathcal{C}), T^\perp)$ where $T(\mathcal{C})$ is the tranposition of $\mathcal{C}$ with respect to the tracker T and $T^\perp$ is the dual tracker.

One key point is to show that the bechet transformation is involutive. The second point is to show that if $\mathcal{C}$ is minimaly cyclic and T is a cyclic tracker of $\mathcal{C}$, then $Bechet(\mathcal{M}, \mathcal{C}, T)$ corresponds to a module that contains no switching cycle.

9.1.1 Bechet transformation of pre–modules

In this subsection we work on graphs that are *pre–modules* see the definition 5.1. For clarity let us recall the notions of input and output of a pre–module $\mathcal{M}$;

$$Input(\mathcal{M}) = \{\alpha \in Arr_{\mathcal{M}} \mid \mathsf{src}.\alpha = \bot\}$$
$$Output(\mathcal{M}) = \{\alpha \in Arr_{\mathcal{M}} \mid \mathsf{tgt}.\alpha = \bot\}$$

Configuration of a Pre–module. Given a pre–module $\mathcal{M}$, we call *configuration* of $\mathcal{M}$ any set of disjointed pairs of input arrow of $\mathcal{M}$. Given a pre–module $\mathcal{M}$ and $\mathcal{C}$ one of its configurations. We will denote by $\mathcal{C}(\mathcal{M})$ the module obtained, after linking each paired input by $\mathcal{C}$ by an axiom node. Let us give a formal definition;

- $Arr_{\mathcal{M}} = Arr_{\mathcal{C}(\mathcal{M})}$ The arrows remain the same.
- $tgt_{\mathcal{M}} = tgt_{\mathcal{C}(\mathcal{M})}$ For each arrow the target is unchanged.
- $V_{\mathcal{C}(\mathcal{M})} = V_{\mathcal{M}} \cup \{n_p \mid p \in \mathcal{C}\}$ We add for each pair of $\mathcal{C}$ a node denoted n_p.
- $src_{\mathcal{C}(\mathcal{M})}(\alpha) = src_{\mathcal{M}}(\alpha)$ if α is not paired by $\mathcal{C}$, ie. $\alpha \notin \bigcup \mathcal{C}$. But otherwise $\alpha \in p$ with $p \in \mathcal{C}$, in this case $src_{\mathcal{C}(\mathcal{M})}(\alpha) = n_p$ one of the new node introduced.

- Each node n_p is labeled by an axiom.

The point of a configuration of a pre–module $\mathcal{M}$ is to describe how axiom nodes could be added to a module $\mathcal{M}$. The goal is to study not proof structures, but merely their bodies $\mathcal{B}_S$ and their configurations. In the case of proof structures without cuts and with one conclusion, it therefore corresponds to the study of $\otimes, \parr$ trees and their configurations.

A configuration $\mathcal{C}$ will be said to be *well-typed*[38] if given any pair of arrows $\{\alpha, \beta\}$ of $\mathcal{C}$, α is labeled by some formula A, while β is labeled by its linear negation $A^\perp$.

Definition 9.1 (Tracker)**.** Given a premodule $\mathcal{M}$ and one of its configuration $\mathcal{C}$. A *tracker* of $\mathcal{C}$ is a triplet (σ, π_1, π_2) made of;

- (Permutation) A cyclic permutation $\sigma : \mathcal{C}_0 \to \mathcal{C}_0$ of a subset $\mathcal{C}_0 \subset \mathcal{C}$ with no invariant.[39] $\mathcal{C}_0$ is called the *domain* of T.
- (Polarity) $\pi_1, \pi_2 : \mathcal{C} \to \bigcup \mathcal{C}$ two choice function for the pairs of $\mathcal{C}$, such that $\pi_1(p) \neq \pi_2(p)$ for all pairs of $\mathcal{C}$. The pair (π_1, π_2) might be called the *polarity* of T.

Transposed configuration. Given a pre–module $\mathcal{M}$, one of its configuration $\mathcal{C}$, and a tracker $T = (\sigma, \pi_1, \pi_2)$ of $\mathcal{C}$. We define the transposed configuration of $\mathcal{C}$ relatively to T, that we denote $T(\mathcal{C})$ as the set of pairs

$$T(\mathcal{C}) = \{\{\pi_2(p), \pi_1(\sigma(p))\} \mid p \in \mathcal{C}\}$$

We will want to proove that given a minimaly cyclic configuration $\mathcal{C}$ of a tree $\mathcal{T}$, any transposition $T(\mathcal{C})$ is acyclic in dual tree $\mathcal{T}^\perp$, when T is a cyclic tracker of $\mathcal{C}$.

Dual Tracker. Given a premodule $\mathcal{M}$, $\mathcal{C}$ a configuration of $\mathcal{M}$ and a tracker T of $\mathcal{C}$. We define the *dual tracker* of T denoted $T^\perp$ as a tracker of $T(\mathcal{C})$, such that we define for a pair p' of $T(\mathcal{C})$ [40]

- $\pi_1^\perp(p') = \pi_2(p)$ for each $p' \in T(\mathcal{C})$.
- $\pi_2^\perp(p') = \pi_1(\sigma(p))$ for each $p' \in T(\mathcal{C})$.
- $\sigma^\perp$ sends $p' = \{\pi_2(p), \pi_1(\sigma(p))\}$ to $\{\pi_2(\sigma(p)), \pi_1(\sigma\sigma(p))\}$.

Béchet Transformation. Given a pre–module $\mathcal{M}$ we define the bechet tranformation as it follows, it takes a pair $(\mathcal{C}, T)$ of a configuration of $\mathcal{M}$ and

[38] We first study acyclicity and cyclicity without care of well-typedness. In fact the point of the instantiation ϕ introduced later by Béchet is to ensure that we can construct from an acyclic configuration a well-typed acyclic one.

[39] It indicates in which cyclic order we must travel, in particular to find cycles in modules. Also, $\mathcal{C}_0$ may eventually be $\mathcal{C}$.

[40] Recall that these pairs are of the form $\{\pi_2(p), \pi_1(\sigma(p))\}$ where $p \in \mathcal{C}$.

a tracker of this configuration and sends it the pair $(T(\mathcal{C}), T^\perp)$ of the same nature[41]. Our next step is to show that this transformation is involutive.

Intuitively the transposed configuration corresponds to a repairing of the input arrows, such that we pair the arrows that (if T tracks a cycle) meets according to the tracker T.

Proposition 9.1 (Involution of the Bechet transformation)**.** *Given a configuration $\mathcal{C}$ of some module $\mathcal{M}$, and one of its tracker T, we have $T^\perp(T(\mathcal{C})) = \mathcal{C}$.*

Proof. Consider the pairs of the transposed configuration $T(\mathcal{C})$, they are of the form $d = \{\pi_2(p), \pi_1(\sigma(p))\}$ where p range over $\mathcal{C}$. On the other hand the pairs of $T^\perp(T(\mathcal{C}))$ are of the form $\{\pi_2^\perp(d), \pi_1^\perp(\sigma^\perp(d))\}$. And by definition;

- $\pi_2^\perp(d) = \pi_1(\sigma(p))$
- $\pi_1^\perp(\sigma^\perp(d)) = \pi_1^\perp(\{\pi_2(\sigma(p), \pi_1(\sigma\sigma(p)))\}) = \pi_2(\sigma(p))$.

Therefore the element of $T^\perp(T(\mathcal{C}))$ are the pairs $\{\pi_1(\sigma(p)), \pi_2(\sigma(p))\}$, where $p \in \mathcal{C}$. Since σ is a permutation, it follows that $T^\perp(T(\mathcal{C})) = \mathcal{C}$.

Now let us show that $T^{\perp\perp}$ corresponds to T. Consider a pair $p = \{\pi_2^\perp(p_0), \pi_1^\perp(\sigma^\perp(p_0))\}$;

- $\pi_1^{\perp\perp}(p) = \pi_2^\perp(p_0) = \pi_1(p_0)$

□

9.1.2 Bechet transformation with modules, effects on cyclicity

We want to use the introduced notion on pre–modules of configuration to study the cyclicity and the acyclicity of MLL modules.

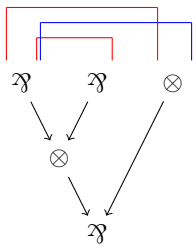


Figure 25: A $\otimes, \parr$-tree, and two of its configuration, a cyclic and an acyclic one.

Definition 9.2 (Canonical configuration)**.** Given a proof structure S. We define its *canonical configuration* that we denote $\mathcal{C}_S$ as;

$$\mathcal{C}_S = \{\{i, j\} \mid \exists u \in V_S^{(ax)}(src(i) = src(j) = u)\}.$$

Proposition 9.2 (Proof structures as configurated trees)**.** *Given a proof structure S, we have the identity $S = \mathcal{C}_S(\mathcal{B}_S)$ (in the sense of isomorphism).*

[41] Note that $T(\mathcal{C})$ is also a configuration of $\mathcal{M}$, and $T^\perp$ is a tracker of this configuration.

Proof. We show that these two structures are equal using the notion of isomorphism.

By definition the body preserves the arrows $Arr(\mathcal{B}_S) = Arr(S)$ and even their targets; $tgt_{\mathcal{B}_S} = tgt_S$. Since the application of configuration, also does not affect arrows and their target we can conclude; $Arr(\mathcal{C}_S(\mathcal{B}_S)) = Arr(S)$ and $tgt_{\mathcal{C}_S(\mathcal{B}_S)} = tgt_S$. It remains to show the correspondancy between the set of vertices and the source function.

By definition $V_{\mathcal{B}_S} = V_S \setminus V_S^{(Ax)}$, and $\mathcal{C}_S(\mathcal{B}_S) = V_{\mathcal{B}_S} \cup \{n_p \mid p \in \mathcal{C}\}$ Now consider the following function $f : V \to V_{\mathcal{C}(\mathcal{B}_S)}$, which given $x \in V$ works the following way;

- if $x \notin V_S^{(Ax)}$, then $f(x) = x$.
- If $x \in V_S^{(Ax)}$, then x is the source of a pair of arrow $p = \{\alpha, \beta\}$ and $f(x) = n_p$.

To show the equality it remains only to show that for each arrow $f(src_S(\alpha)) = src_{S'}(\alpha)$. Consider some arrow α of $Arr(S)$;

- If α has its source in a that is not an axiom, then $f(src_S(\alpha)) = src_S(\alpha)$. And more, α has the same source in S and in $\mathcal{B}_S$ and therefor also in $\mathcal{C}_S(\mathcal{B}_S) = S'$.
- If the source of α is an axiom node u in S, then $p = \{\alpha, \beta\} \in \mathcal{C}_S$ for some arrow β. Then in $\mathcal{B}_S$ the source of α is empty. While in $\mathcal{C}_S(\mathcal{B}_S)$ the source of α corresponds to n_p. But by definition of f, $f(u) = n_p$ so, $f(src_S(\alpha)) = src_{S'}(\alpha)$.

This show that $S = \mathcal{C}(\mathcal{B}_S)$. With the isomorphism f for vertices, and id for the arrows. □

Switching cycles and configurations. Given a module $\mathcal{M}$. We can question whether a configuration generates cycles or not. Meaning if we add axioms according to a given configuration, will it result in a module that has a switching cycle. To do so we use the notions of paths and meeting points as seen in the section 3.

Given a configuration $\mathcal{C}$ we will say it is *cyclic*, if the module obtained after completion $\mathcal{CM}$ contains a switching cycle. Calling the bechet's cyclicity criterion ie. the theorem 4.1, we can characterize the cyclicity of some configuration $\mathcal{C}$, by the existence of some enumerations such that the arrows α_i, β_i correspond to a pair $p_i \in \mathcal{C}$.

We will say that a tracker is *cyclic* or that it is *tracking a cycle* in $\mathcal{C}$ if the following holds;

- For each p in the domain of σ, $\pi_2(p)$ and $\pi_1(\sigma(p))$ meet on nodes that are either labeled by **cut** or $\otimes$.
- All the meeting points of the pairs of $\{(\pi_2(p), \pi_1(\sigma(p))) \mid p \in dom(\sigma)\}$ are distinct.

With these notions, we want to show that configuration $\mathcal{C}$ is *cyclic*, if and only if, it has cyclic tracker. A configuration that is not cyclic will be said to be *acyclic*. Now we want to study how the property of cyclicity and acyclicity of configurations behave with dual trees.

Switching cycles and polarity. We want to express here why we do not define more simply configurations as ordered pairs. The choice of a tracker to contain the two functions of polarity π_1, π_2 does have an importance. See the exemple in figure 26. The point is that while $\pi_2(p)$ and $\pi_1(\sigma(p))$ meet on a tensor, it could be that $\pi_1(p)$ and $\pi_1(\sigma(p))$ meet on a parr node. Therefor the choice function π_1, π_2 is not without consequence, for the same permutation, a change of polarity can make a tracker acyclic in the sense of the previous definition.

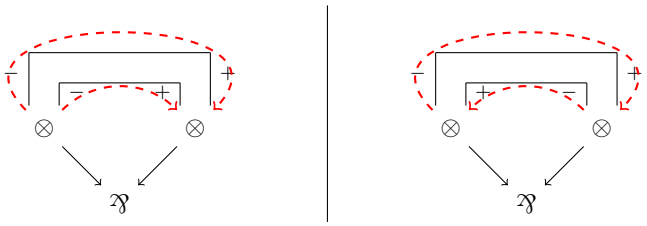


Figure 26: Cyclicity and polarity.

The previous figure is there to stress the importance of the ordering of the pairs (which could be seen as polarity). With the idea of polarity, the negative (π_1 or $-$) is compatible with the positive (π_2 or $+$), while two arrow with the same polarity are not compatible. In the first figure there is no cycle for a tracker with the wrong polarization π_1, π_2, since we have two pairs such that the negative and positive arrows of these pairs always meet on the root that is a $\parr$ node. In the second figure, with this polarization indeed we can find a cyclic tracker T, actually taking for σ any of the two transposition of $\mathcal{C}$. This holds since the negative and positive arrows of the two pairs of $\mathcal{C}$ always meet on an $\otimes$ node, and the meeting points are distincts.

We choose not to define a configuration as a partial function since it would induce a natural polarization (antecedent - image) that could be wrong. Also there does not seem to be, such a thing as a canonical polarization since, one polarization could reveal one cycle, while another one could reveal a second cycle that was not revealed by the first. For each cycle corresponds (at least one) a polarity, and a permutation, meaning a tracker.

Proposition 9.3 (characterization of cyclicity by trackers)**.** *Given a (cut free) proof structure S. The two statement are equivalent;*

1. *S has a switching cycle.*
2. *The canonical configuration $\mathcal{C}_S$ of $\mathcal{B}_S$ has a cyclic tracker.*

Proof. **(1⇒2).**Assume S has a switching cycle, calling Bechet's cyclicity criterion, we can ensure there exists $N_1, ..., N_k$ axiom nodes in S, and arrows $\{(\alpha_i, \beta_i)\}_{1 \leq i \leq k}$. Such that each β_i, α_{i+1} meet on distinct nodes.

This nodes are labeled either by $\otimes$ or **cut** nodes. Since if they were label by a $\parr$ the cycle would not be a switching cycle. But also it cannot be a cut node since S is supposed to be cut free.

But then note that each pairs $\{\alpha_i, \beta_i\}$ are by definition[42] elements of the canonical configuration $\mathcal{C}_S$ since they are the two output arrows of the same node (an axiom). Consider the permutation σ of $\mathcal{C}$ defined on $\bigcup\{\alpha_i, \beta_i\} \subset \mathcal{C}_S$, and that sends each $\{\alpha_i, \beta_i\}$ to $\{\alpha_{s(i)}, \beta_{s(i)}\}$, where s denotes the successor permutation for $\{1, ..., k\}$.

We then define $\pi_1(\{\alpha_i, \beta_i\}) = \alpha_i$, while π_2 returns β_i. We can then easily ensure that each $\pi_2(p)$ and $\pi_1(\sigma(p))$ meet in distincts $\otimes$ nodes, simply by equalities.

(2⇒1). The construction is the reverse, indexing the support $supp(\sigma)$ as $p_1, ..., p_n$. The nodes are defined as $N_i = src.(\pi_1(p_i))$. Then we define $\alpha_i = \pi_1(p_i)$, while $\beta_i = \pi_2(p_i)$. □

Minimal cylicity. We say that a configuration $\mathcal{C}$ is *minimaly cyclic* if it is cyclic and if by removing any its pairs p, we obtain $\mathcal{C} \setminus \{p\}$ an acyclic configuration. equivalently it means there is no cyclic tracker $T = (\sigma, \pi_1, \pi_2)$ of domain $\mathcal{C}_0$ of $\mathcal{C}$. This is straightforward to show, since if it was not the case, we could remove from $\mathcal{C}$ a pair p that is not in $\mathcal{C}_0$. T would still correspond to a cyclic tracker in $\mathcal{C} \setminus \{p\}$ since its domain is included in $\mathcal{C}_0$. On the other hand, if there is no cyclic tracker of $\mathcal{C}$, remark that a cyclic tracker of domain $\mathcal{C} \setminus \{p\}$ is indeed a cyclic tracker for $\mathcal{C}$ (since the domain is included in $\mathcal{C}$).

Cyclicity and duality. Let us study how the cyclicity of a configuration of some module $\mathcal{M}$, behaves relatively to duality. It can be tempting to say that if $\mathcal{C}$ is cylic in $\mathcal{M}$ then $\mathcal{C}$ is acyclic in the dual $\mathcal{M}^\perp$. While this would be true for minimal 1-cyclic[43] configurations, it already fails for 2-cyclic configuration [44] see the figure 29.

Although the argument of Bechet in his proof of lemma 15 of the original article [BEC98], uses at least the fact that if a $\otimes, \parr$ tree $\mathcal{T}$ has a (minimaly) cyclic configuration then $\mathcal{T}^\perp$ has a (minimaly) acyclic one. In fact Bechet give's a rather 'canonical' way to transform a cyclic configuration of $\mathcal{T}$ in an acyclic one of $\mathcal{T}^\perp$ which is the one we exposed in the previous subsection on pre–modules.

[42] They are pairs of arrows with a common source, labeled by an axiom.

[43] Given an integer n, a n-cyclic configuration is a configuration that has cyclic tracker $T = (\sigma, \pi_1, \pi_2)$ with $card(supp(\sigma)) = n$.

[44] and so even for some non minimal 1-cyclic configurations

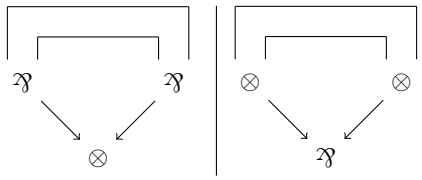


Figure 27: The cyclicity of a configuration may be conserved in the dual

The same thing happens with acyclicity, which may be conserved, see the next figure. Obviously the two proof structures are acyclic since they correspond to proof nets of a sequent $\vdash A, A^{\perp}$.

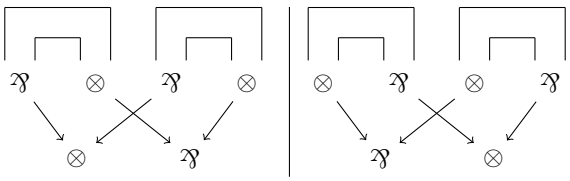


Figure 28: The acyclicity of a configuration may be conserved in the dual

Bechet's transformation, cyclicity and duality. Given a module $\mathcal{M}$, we will now want to explore how the bechet transformation of a cyclic configuration and its tracker $(\mathcal{C}, T)$ into $(T(\mathcal{C}), T^{\perp})$ affects the cyclicity in the dual modules $\mathcal{M}^{\perp}$.

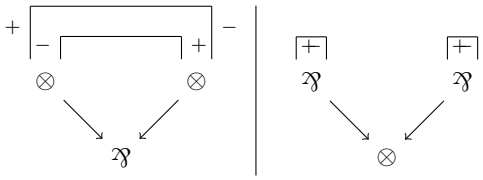


Figure 29: The transposition of a minimaly cyclic configuration, it becomes acyclic in the dual

We will expose here an implication from minimal cyclicity to acyclicity, in the annexe B we explore the possibility for an equivalency (which for the argument of Béchet is not necessary).

Lemma 9.1 (The transposition of a minimaly cyclic configuration is acyclic in the dual.)**.** *Given a module $\mathcal{M}$ of labels $\otimes$ and $⅋$ and $\mathcal{C}$ a configuration of $\mathcal{M}$, the two statement are equivalent;*

If $(\mathcal{C}, T)$ is a minimaly cyclic configuration of $\mathcal{M}$ then its béchet transformation $T(\mathcal{C})$ is an acyclic configuration in the dual $\mathcal{M}^{\perp}$.

Proof. **(1⇒2).** Given the configuration $\mathcal{C}$ and its tracker T, let's us show that the transposition $T(\mathcal{C})$ generates no switching cycles.

Ad absurdum, assume that $T(\mathcal{C})$ generates a cycle of length $m \leq n$. Consider therefore the tracker (μ, π'_1, π'_2) of this cycle.

Consider $d_1, ..., d_m$ an enumeration of the support of μ. Then consider the collection of arrows, $\mathcal{F} = \{\{\pi'_1(d_k), \pi'_2(d_k)\} \mid 1 \leq k \leq m\}$. Then let's explicit two collections of meeting points;

- We know that each pair $(\pi'_1(d_k), \pi'_2(d_k))$ meet in distinct $\parr$'s in $\mathcal{M}^\perp$ ensuring m meeting points. This is because $d_k = \{\pi_2(p), \pi_1(\sigma(p))\}$ and by assumption on the tracker T to be cyclic, we know that these pairs of arrows meet in distinct $\otimes$ in $\mathcal{M}$.
- On the other hand, assuming the tracker (μ, π'_1, π'_2) describes a cycle means by definition $(\pi'_2(d_k), \pi'_1(\mu(d_k)))$ meet in distincts $\otimes$ nodes, ensuring m meeting points labeled by $\otimes$ in $\mathcal{M}^\perp$.

But note that none of these two collections of m nodes cannot overlap, since one corresponds to nodes labeled by $\parr$'s while the other one corresponds to nodes labeled by $\otimes$ (in $\mathcal{M}^\perp$). This ensure that the family $\mathcal{F}$ has $2m$ meeting points. But this goes against the upper bound of meeting points, the proposition 4.10 ensures that there is at most $2m - 1$ meeting points for a family of $2m$ arrows. We have a contradiction, therefore the transposition $T(\mathcal{C})$ does not generate any cycle of length m.

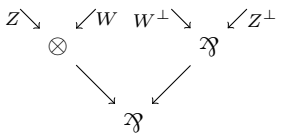
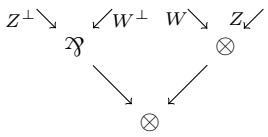
□

9.2 Correctness of the Bechet transformation

Having shown that given an adapted triplet $(\mathcal{M}, \mathcal{C}, T)$ its Bechet transformation $(\mathcal{M}^\perp, T(\mathcal{C}), T^\perp)$ correspond to a pre proof structure $T(\mathcal{C})[\mathcal{M}^\perp]$ with no switching cycles, but record that the obtained graph is not necessarily well typed. The point is to show that we can from this transformation, obtain a well typed one, this is the point of the instantiation ϕ introduced by bechet, in fact it allows to transform any configuration into a well typed configuration.

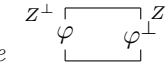
An Instantiation to generate well typedness. Let's denote by ϕ the substitution of atomic formulas that instantiate for any given $X \in \mathcal{V}$, X by $\phi(X) = (Z \otimes W) \parr (W^\perp \parr Z^\perp)$ and $X^\perp$ by $\phi(X^\perp) = (Z^\perp \parr W^\perp) \otimes (W \otimes Z)$.

We will denote their respective module representations φ and $\varphi^\perp$ meaning that

$\varphi =$  and $\varphi^\perp =$ 

Remark that φ can always be connected to $\varphi^\perp$ connecting the $Z^\perp$ input of φ to the Z input of $\varphi^\perp$, resulting in the module that we will denote $\varphi \overset{\ulcorner}{}_{Z^\perp}\!\!\!\!\!\!\underset{Z}{\urcorner}\ \varphi^\perp$. Now we pretend that linking the conclusion of the obtained graph by a cut node produces a bad proof structure.

Given a module $\mathcal{M}$ and an instatiation I of formulas, $I(\mathcal{M})$ denotes the module obtained after relabeling each input arrow (denoting its current label A) that either has an empty source, or has an axiom as a source, we label the arrow by $I(A)$.

Proposition 9.4 (Deadlock-reduction)**.** *The proof structure*  *reduces to a proof structure containing a deadlock, and so is, a bad proof structure.*

Proof. We merely apply cut elimination. □

Development of a module. Given a module $\mathcal{M}$, we define it's development that we donete $dev(\mathcal{M})$ as the module corresponding to $\mathcal{M}$ such that each input α is composed with the only output of the canonical representation module $[\![A]\!]$ where A is the formula labeling α.

Well typed configuration and the instantiation ϕ. Recall that configurations $\mathcal{C}$ are – in general – not necessarily well typed. And even worse for us, the fact that $\mathcal{C}$ is is well typed, does not ensure at all that a transposition $T(\mathcal{C})$ is well typed. Therefore the operation of transposition has to involve an instantiation – which here will be the instantiation introduced previously ϕ – to become well typed.

Given a configuration $\mathcal{C}$ of some module $\mathcal{M}$. We can define a well typed configuration corresponding to $\mathcal{C}$ for the developped module $dev(\phi(\mathcal{M}))$. Think that each input arrow α of $\mathcal{M}$, is in the module $dev(\phi(\mathcal{M}))$connected (ie. in the arrangement) of both an arrow labeled by Z denoted α_Z and one labeled by $Z^\perp$ denoted $\alpha_{Z^\perp}$.

Then we define the instanted configuration $\phi(\mathcal{C})$ as the pairs $\{\alpha_Z, \beta_{Z^\perp}\}$ given a pair $p = \{\alpha, \beta\}$ of $\mathcal{C}$.

$$\phi(\mathcal{C}) = \{\{\pi_1(p)_Z, \pi_2(p)_{Z^\perp}\} \mid p \in \mathcal{C}\}$$

?ote that the arrows paired by $\phi(\mathcal{C})$ can both belong to modules of the form ϕ of $\phi^\perp$, we have no guarantee that one arrow is in a module ϕ while the other is in a module $\phi^\perp$. Since this would mean that $\mathcal{C}$ always pair arrow labeled by a positive formula A to an arrow labeled by a negative one $B^\perp$. This, if $\mathcal{C}$ isn't well typed is not guaranteed.

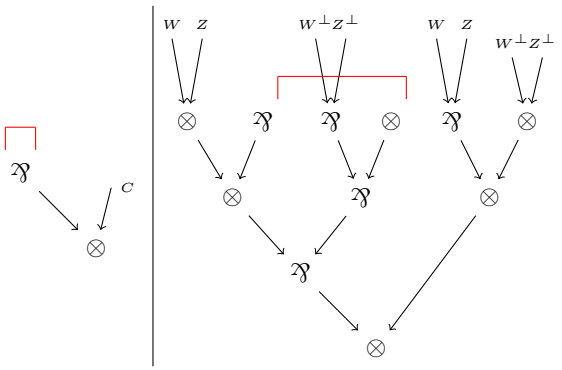


Figure 30: A configuration $\mathcal{C}$ for $\mathcal{T}$, and its associated configuration $\phi(\mathcal{C})$ in the tree $dev(\mathcal{T})$

Proposition 9.5 (Instantiated configurations preserve acyclicity)**.** *Given a module $\mathcal{M}$, and one of its configuration $\mathcal{C}$.*

If $\mathcal{C}$ is an acyclic configuration, then $\phi(\mathcal{C})$ is an acyclic configuration of $dev(\phi(\mathcal{M}))$ that is well typed.

Proof. Consider a tracker of $T = (\sigma, \pi_1, \pi_2)$ of $\phi(\mathcal{C}))$, let us show that T cannot track any cycle in $dev(\phi(\mathcal{T})) := \mathcal{M}'$. If not, consider the support of σ then for each $p \in supp(\sigma)$ we'd have $\{\pi_2(p), \pi_1(\sigma(p))\}$ meeting in distincts $\otimes$ node.

But then consider an input arrow of $\mathcal{M}'$. Necessarily this arrow is simultaneously an input arrow of a submodule of $\mathcal{M}'$ that corresponds to the development of α, ie. $dev(\alpha)$, where α is an input of $\mathcal{M}$.

Therefore given a pair $\{\beta_1, \beta_2\} = \{\pi_2(p), \pi_1(\sigma(p))\}$ the two arrows cannot be a part of the same submodule $dev(\alpha)$ since this would mean $\mathcal{C}$ has two pairs with α.

Therefore β_1 attains[45] an input arrow α_1 while β_2 attains an input α_2. And the two path are distinct. Therefore if β_1, β_2 meet in distinct $\otimes$, then this will also be true for α_1, α_2. And so it will induce a cycle in $\mathcal{C}$, which goes against the assumptions.. □

Proposition 9.6 (Generating a correct module)**.** *Given a module $\mathcal{M}$ and $\mathcal{C}$ an acyclic configuration of $\mathcal{M}$.*

Then $\phi(\mathcal{C})[dev(\phi(\mathcal{M}))]$ is correct module.

Proof. From the previous proposition we know that $\phi(\mathcal{C})$ is well typed, meaning $\mathcal{M}' = \phi(\mathcal{C})[dev(\phi(\mathcal{M}))]$ is a proof structure. Let's now consider $\sigma\mathcal{M}'$ on of its switchin and show it is acyclic and has no internal component.

Still from the previous proposition, given that $\mathcal{C}$ is acyclic, $\phi(\mathcal{C})$ is also an acyclic configuration, ensuring $\sigma\mathcal{M}'$ does not have any cycles.

[45] Meaning there is path, ie. a sequence of arrow starting by β_1 and finishing by α_1

Let's show it does not have any internal component, meaning each node is related (ie. is in the track) to an input or an output arrow. Consider u a node of $\sigma\mathcal{M}'$, note that $V_{\mathcal{M}'} = V_{\mathcal{M}} \cup \bigcup_{\alpha \in Input(\mathcal{M})} V_{dev(\alpha)}$;

- If u is not a node of $\mathcal{M}$, then u is a node of a submodule ϕ or $\phi^\perp$, but note that any node of ϕ or $\phi^\perp$ is after switching related to an output labeled by W or $W^\perp$. This remains true in $\mathcal{M}'$ since $\phi(\mathcal{C})$ pairs only arrow labeled by Z or $Z^\perp$.
- If u is a node of $\mathcal{M}$, then correctness of $\mathcal{M}$ ensure that in $\sigma|\ \mathcal{M}\mathcal{M}\ u$ is related to an output or an input of $\mathcal{M}$. If it is related to an output arrow then since the application of a configuration does change outputs, it remains true in $\mathcal{M}'$.
- If on the other u is related to an input arrow α of $\mathcal{M}$, then this arrow α is the conclucion of a development of its label, either $\phi(X)$ or $\phi(X^\perp)$, ie. either ϕ or $\phi^\perp$, so this arrow is related to an input arrow labeled by W or $W^\perp$ in $\sigma\mathcal{M}'$.

The module has no internal component nor cycles after switching therefore it is correct. □

9.3 Normalizing switching cycles into deadlock

For this part since it consists of a process of cutelimination we recall the section 7.2. We will describe the reduction to a deadlock into this formal framework. Recalling the section 7.2, we want to stress that being given a module without cut and axioms is being merely given a forest

$$[\xi_1, U_1 : A_1], ..., [\xi_n, U_n : A_n].$$

The notion of configuration that we presented on inputs, here is more formal since it consists of disjoint pairs of $\bigcup U_i$. Finally to involve cuts is to involve disjoint pairs $\{\xi_1, ..., \xi_n\}$.

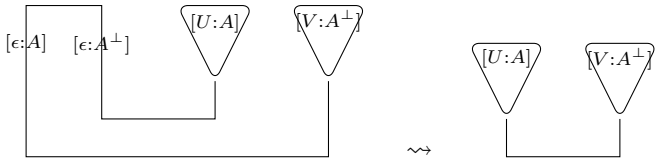


Figure 31: Illustration of The detour elimination

Proposition 9.7 (Detour elimination)**.** *Given a formula A we call* detour *a module*

$$\frac{\{\{\nu, \xi'\}\}}{[\xi, U : A] \times [\nu, \epsilon : A^\perp] + [\xi', \epsilon : A] \times [\nu', V : A^\perp]}$$

such a module reduces to

$$\frac{[\nu, U : A], [\xi', V : A^\perp]}{\{\{\nu, \xi'\}\}}$$

Proof. Note that $\dfrac{\{\{\nu,\xi'\}\}}{[\xi,U:A]\times[\nu,\epsilon:A^\perp]+[\xi',\epsilon:A]\times[\nu',V:A^\perp]}$ has for submodule the following,

$$\frac{\{\{\nu,\xi'\}\}}{[\xi,U:A]\times[\nu,\epsilon:A^\perp],[\xi',\epsilon:A]} = \frac{\dfrac{\{\{\nu,\xi'\}\}}{[\xi,U:A],[\nu,\epsilon:A^\perp],[\xi',\epsilon:A]}}{\{\{\xi,\nu\}\}}$$

by definition this reduces (case of normal cut) to $[\xi',\epsilon\setminus\epsilon\cup U:A]$ meaning the tree $[\xi',U:A]$. Therefore by locality,

$$\frac{\{\{\nu,\xi'\}\}}{[\xi,U:A]\times[\nu,\epsilon:A^\perp]+[\xi',\epsilon:A]\times[\nu',V:A^\perp]}$$

reduces to the module $[\xi',U:A]\times[\nu',V:A^\perp]$.

But note that ξ,ξ',ν,ν' are distinct therefore we can rename the relative adress ν' in ν so the two modules are equivalent;

$$[\xi',U:A]\times[\nu',V:A^\perp]\equiv[\xi',U:A]\times[\nu,V:A^\perp]$$

Since cut elimination is compatible with the equivalency, it follows that the original module reduces to $[\xi',U:A]\times[\nu,V:A^\perp]$ which denotes the module;

$$\frac{\{\{\nu,\xi'\}\}}{[\xi',U:A],[\nu,V:A^\perp]}$$

□

Partial reduction, and ordering partitions. Given V,W two set of occurences, we use the notation $V\ll W$, to mean that for any sequence $v\in V$ there exists some sequence $w\in W$ such that v is a subsequence of w. We will also say that a module $\mathcal{M}$ partially reduces to a module $\mathcal{M}'$ if, $\mathcal{M}$ reduces to a module $\mathcal{M}_0$ and $\mathcal{M}'$ is a submodule of $\mathcal{M}_0$. We might denote it by $\mathcal{M}\rightsquigarrow^\bullet\mathcal{M}'$

Proposition 9.8 (Multiplicative cut–elimination horizontal locality)**.** *Given two tree $\mathcal{T}_1=[U:A]$ and $\mathcal{T}_2=[V:A^\perp]$, two relative adresses ξ and ν. Assuming we are given another partition W of the formula A such that $V\ll W$, then if the following reduction is true,*

$$\xi\mathcal{T}_1\times\nu\mathcal{T}_2\rightsquigarrow_m\mathcal{M}_1\quad\textit{where}\quad\xi\mathcal{T}_1(u)\times\nu\mathcal{T}_2(u)\subset\mathcal{M}_1$$

so is the following reduction,

$$\xi\mathcal{T}_1\times\nu\mathcal{T}_2'\rightsquigarrow_m\mathcal{M}_2\quad\textit{where}\quad\xi\mathcal{T}_1(u)\times\nu\mathcal{T}_2'(u)\subset\mathcal{M}_2$$

Proof. This is rather straightforward we reason by induction on the number of reduction step but note that showing it holds for one step of reduction ensures it for any quantity of steps. Assuming U and V both non empty (so that the reduction can be done). $[\xi, U : A] \times [\nu, V : A^\perp]$ reduces to $[\xi 1, U^{(1)} : A/1] \times [\nu 1, V^{(1)} : A^\perp/1] + [\xi 2, U^{(2)} : A/2] \times [\nu 2, V^{(2)} : A^\perp/2]$

Now if $V \ll W$ then W is still non empty (so reduction can be applied still) and by definition; $[\xi, U : A] \times [\nu, W : A^\perp]$ reduces to $[\xi 1, U^{(1)} : A/1] \times [\nu 1, W^{(1)} : A^\perp/1] + [\xi 2, U^{(2)} : A/2] \times [\nu 2, W^{(2)} : A^\perp/2]$ □

Deadlock. In this framework a dead lock corresponds to an expression

$$\dfrac{\dfrac{\{\{\xi\nu\}\}}{[\xi, \epsilon : A], [\nu, \epsilon : A^\perp]}}{\{\{\xi, \nu\}\}}$$

where ξ, ν are some relative adresses, and A is some MLL formula.

Proposition 9.9 (Generalized Deadlock)**.** *Given an MLL formula A, the module*

$$\dfrac{\{\{\nu_i, \xi_{\sigma(i)}\} \mid 1 \leq i \leq n\}}{\sum\limits_{1 \leq i \leq n} [\xi_i, \epsilon : A] \times [\nu_i, \epsilon : A]}$$

reduces to a deadlock. Where σ is a permutation without invariant or with only 1 *antecedant.*

Proof. By Induction. We reason by induction on the length n. If $n = 1$ then the sum corresponds merely to a deadlock (noting that σ can only be the identity in this case).

Now let us assume $\dfrac{\{\{\nu_i, \xi_{\sigma(i)}\} \mid 1 \leq i \leq n\}}{\sum\limits_{1 \leq i \leq n} [\xi_i, \epsilon : A] \times [\nu_i, \epsilon : A]}$ reduces to a deadlock (for any family of size n and permutation σ). Let's show it is still true for a size $n + 1$.

$$\dfrac{\{\{\nu_i, \xi_{\sigma(i)}\} \mid 1 \leq i \leq n+1\}}{\sum\limits_{1 \leq i \leq n+1} [\xi_i, \epsilon : A] \times [\nu_i, \epsilon : A]}$$

Consider the following submodule $\dfrac{\{\nu_1, \xi_2\}}{[\xi_1, \epsilon : A] \times [\nu_1, \epsilon : A^\perp] + [\xi_2, \epsilon : A] \times [\nu_2, \epsilon : A^\perp]}$. Now calling the previous proposition we can ensure this reduces to $[\xi_2, \epsilon : A] \times [\nu_2, \epsilon : A^\perp]$ (eventually using equivalency).

But so this means after this step of reduction we obtain a sum made of n opposition that is,

$$\frac{\{\{\nu_i, \xi_{\sigma(i)}\} \mid 2 \leq i \leq n+1\}}{\sum\limits_{2 \leq i \leq n+1} [\xi_i, \epsilon : A] \times [\nu_i, \epsilon : A]}$$

we can therefore conclude by induction, this sums reduces to a deadlock since it is made of n elements. □

Instantiation. The notion of instantion can be defined in trees given a formula $[U : A]$ the instantiation $\phi([U : A])$ consists of $[U : \phi(A, U)]$ where $\phi(A)$ is a formula defined by induction as;

- If $U = \epsilon$ then $\phi(A, \epsilon) = \phi(A)$
- If U is non empty and $A = F \square G$ then $\phi(A, U) = \phi(F, U^{(1)}) \square \phi(G, U^{(2)})$ where $\square \in \parr, \otimes$.

The point is that given a MLL tree $\mathcal{T}$ cut and axiom free, of conclusion A, then this tree corresponds to $[U : A]$ for some partition of A. The application of an instantiation to $\mathcal{T}$ corresponds to the same tree (so the same partition) but with a different type, namely $[U : \phi(A, U)]$.

Concatenation. Given two sets of occurences U and V, meaning, sets of sequences of 1 and 2, we will denote by $U.V$ – or simply UV if there is no ambiguity – the set of *concatenation* of U and V. Meaning it corresponds to the set of sequences w such that there exists $u \in U$ and $v \in V$ and such that $w = u.v$. Meaning,

$$U.V = \{w \in \{1, 2\}^* \mid \exists u \in U \exists v \in V w = uv\}.$$

For the sake of consision, we will denote by $U + 1$ the concatenation of U and $\{1, 2\}$, and so on for $U + 2 = (U + 1) + 1$ etc...

We want to stress that given a cut free module X, it's bechet transformation corresponds to Y.

Proposition 9.10 (A Dual confrontation that reduces to deadlocks)**.** *Given $\mathcal{C}$ a configuration of the partition U, a subset $\mathcal{C}_0 \subset \mathcal{C}$ and T some tracker of $\mathcal{C}_0$. Given $\mathcal{T} = [U : \phi(A)]$*

Then the module $\dfrac{\xi\mathcal{C}, \nu T(\mathcal{C}_0)}{[\xi, U : \phi(A)] \times [\nu, U + 2 : \phi(A^\perp)]}$ *will reduce to a deadlock containing module.*

Proof. Let's denote $\mathcal{T} = [U : \phi(A)]$ and $\mathcal{T}^* = [U + 2, \phi(A)^\perp]$. First note that calling the proposition on dual confrontation, the following reduction holds;

$$[\xi, U : \phi(A)] \times [\nu, U : \phi(A)^\perp] \rightsquigarrow \sum_{u \in U} [\xi u, \epsilon : \phi(A)/u] \times [\nu u, \epsilon : \phi(A^\perp)/u].$$

Therefore calling vertical locality we also have the reduction;

$$[\xi, U : \phi(A)] \times [\nu, U + 2 : \phi(A)] \rightsquigarrow \sum_{u \in U} [\xi u, \epsilon : \phi(A)/u] \times [\nu u, \epsilon + 2 : \phi(A^{\perp})/u].$$

So it follows by compatibility of cut elimination with the sum, $\dfrac{\xi\mathcal{C}}{[\xi, U : \phi(A)] \times [\nu, U + 2 : \phi(A^{\perp})]}$ reduces to $\dfrac{\xi\mathcal{C}}{\sum_{u \in U} [\xi u, \epsilon : \phi(A)/u] \times [\nu u, \epsilon + 2 : \phi(A^{\perp})/u]}$.

Note that $\mathcal{C}$ is a total configuration therefore, indexing by U is equivalent to indexing by $\bigcup \mathcal{C}$. So given a polarity (π_1, π_2) we can index the sum by $\mathcal{C}$, and the last sum corresponds to;

$$\frac{\xi\mathcal{C}}{\sum_{p \in \mathcal{C}} \xi\mathcal{T}(\pi_1(p)) \times \nu\mathcal{T}^{*\perp}(\pi_1(p)) + \xi\mathcal{T}(\pi_2(p)) \times \nu\mathcal{T}^{*\perp}(\pi_2(p))}$$

Then note that given a pair p the following module is a detour,

$$\frac{\{\{\xi\pi_1(p), \xi\pi_2(p)\}\}}{\xi\mathcal{T}(\pi_1(p)) \times \nu\mathcal{T}^{*\perp}(\pi_1(p)) + \xi\mathcal{T}(\pi_2(p)) \times \nu\mathcal{T}^{*\perp}(\pi_2(p))}$$

therefore calling the proposition of detour elimination it reduces to

$$\nu\mathcal{T}^{*}(\pi_1(p)) \times \nu\mathcal{T}^{*\perp}(\pi_2(p))$$

So since cut elimination is compatible with the sum, it follows that the previous sum reduces to

$$\frac{\xi\mathcal{C}}{\sum_{p \in \mathcal{C}} \xi\mathcal{T}(\pi_1(p)) \times \nu\mathcal{T}^{*\perp}(\pi_1(p)) + \xi\mathcal{T}(\pi_2(p)) \times \nu\mathcal{T}^{*\perp}(\pi_2(p))}$$

$$\rightsquigarrow \quad \sum_{p \in \mathcal{C}} \nu\mathcal{T}^{*}(\pi_1(p)) \times \nu\mathcal{T}^{*\perp}(\pi_2(p))$$

Now given p a pair of the configuration $\mathcal{C}$ note that;

$$[\nu\pi_1(p), \epsilon + 2 : \phi(A)/u] \times [\nu\pi_2(p), \epsilon + 2 : \phi(A^{\perp})/u] \rightsquigarrow \sum_{d \in \epsilon + 2} g(p, d)$$

$$\text{where } g(p, d) = [\nu\pi_1(p)d, \epsilon : \phi(A)/ud] \times [\nu\pi_2(p)d, \epsilon : \phi(A^{\perp})/ud].$$

So by compatibility with the sum we can conclude the same for the summation, and still by compatility we conclude;

$$\mathcal{M} \rightsquigarrow \frac{\nu T(\mathcal{C}_0)}{\sum_{p\in\mathcal{C}}\sum_{d\in\epsilon+2} g(p,d)}$$

In particular we interest ourselves in the element of $\bigcup T(\mathcal{C}_0)$. Note that T is a tracker so it a triplet $T = (\pi_1, \pi_2, \sigma)$, the element of $T(\mathcal{C})$ are pairs of the form $\{\pi_2(p), \pi_1(\sigma(p))\}$ where p ranges over $\mathcal{C}_0$. Now the element of $T(\mathcal{C})^*$ are indeed of the form $\pi_2(p)d, \pi_1(\sigma(p))d'$ where $p \in \mathcal{C}_0$ and $d, d' \in \epsilon + 2$.

On the other hand given some $d, d' \in \epsilon + 2$ and given some pair p of $T(\mathcal{C})$, the module $[\nu\pi_1(p)d, \epsilon : \phi(A)/\pi_1(p)d] \times [\nu\pi_2(p)d', \epsilon : \phi(A^\perp)/\pi_2(p)d']$ is always contained in the sum. So in particular the previous module contains the module.

Consider the following set of pairs on $\mathcal{C}^* = \{\pi_1(p)d, \pi_2(p)d'\} \mid \pi_1(p)d, \pi_2(p)d' \in T(\mathcal{C})$ such that

$$\frac{\nu T(\mathcal{C}_0)}{\sum_{p\in\mathcal{C}_0^*} [\nu\pi_1(p)d, \epsilon : \phi(A)/\pi_1(p)d] \times [\nu\pi_2(p)d', \epsilon : \phi(A^\perp)/\pi_2(p)d']}$$

Since $\pi_2(p)d'$ is a member of $\bigcup T(\mathcal{C}_0)$ it is paired to $\pi_1(\sigma p)d''$, and so on... Calling the last proposition of generalized deadlock, this reduces to a deadlock. □

9.4 Synthesis

Proposition 9.11. *Given S a proof structure without cuts, with one conclusion. Such that S has a switching cycle.*

Then S is a bad proof structure.

Proof. Let's show that one of the instantiation of S, does not interact well with correct environments (ie. proof nets). Consider one of its instantiation, the instantiation $\phi(S)$, and let's show it's interaction with some (in fact on we will construct) proof nets reduces to a deadlock containing module, ie. a bad proof structure.

Note that S has switching cycle, this means that $\mathcal{B}_S$ has a cyclic configuration $\mathcal{C}_S$, this configuration can be minimalized into a minimaly cyclic configuration $\mathcal{C}$.

$\mathcal{C}$ is minimaly cyclic, so it has a tracker T with no invariant tracking its cycle. Two things follow;

- Calling the lemma 9.1, we can ensure that there exists an acyclic $\mathcal{C}' = T(\mathcal{C})$ configuration for the dual tree $\mathcal{B}_S^\perp$. From this we can ensure calling the proposition 9.6 that the module $\phi(\mathcal{C}')[dev(\phi(\mathcal{B}_S^\perp)]$ is correct. Then calling the completion theorem B.2, we know we can complete this correct module into a proof net $\mathcal{P}$.
- But also, calling the previous proposition 9.10 we can ensure that $\phi(S) = [U : \phi(A)][\mathcal{C}_S]$ interacting with $Bechet(\phi(S)) = [U + 2 : \phi(A^\perp)][T(\mathcal{C})]$, is a module that reduce to a module that contains a deadlock.

Since cut elimination is local, from this two point we can ensure that $\phi(S) \llcorner \mathcal{P}$ reduces to a structure containing a dead lock, ie. a basically wrong proof structure. This shows that S is a bad (meaning it is disruptive or interacts badly) proof structure since one of its instantiation fails modularity. □

Lemma 9.2. *Given a proof structure S without cuts, such that one of its switchings contains a cycle. Necessarily S is a bad proof structure.*

Proof. Calling the proposition 9.11 and 6.1. And noting that a $\parr$ closure conserve the fact that the switching contains a cycle. □

Conclusion

Theorem 9.1 (Béchet, 1998 [BEC98])**.** *Given a proof structure S without cuts. The two statements are equivalent;*

1. *S is incorrect.*
2. *S is a bad proof structure.*

Proof. **(1⇒2).** For this case we assume S incorrect in the sense of Danos-Regnier [DR89], this means one of the two cases occurs, either it has a disconnected switching or it is contains a switching cycle. If S has disconnected switching we call the lemma 8.1. Otherwise if it is containing a switching cycle, we call the lemma 9.2.

In both cases we can ensure that S is a bad proof structure.

(2⇒1). For this case see theorem 6.1 (contraposition). □

MLL correctness is minimal. Therefore we showed that the introduced notion of disruptive (ie. bad, ie. with bad behavior) proof structure, is equivalent to the one of incorrectness, for cut free proof structures. To put it differently, for structures withtout cuts, being correct is equivalent to being well behaved relatively to cut elimination.

Therefore a proof structure that normalizes[46] into a disruptive proof structure, is a proof structure that normalize to an incorrect structure. Since cut elimination preserves correctness, it cannot therefore be a proof net.

Although generally incorrectness does not corresponds to normalization into a disruptive structure, since -in some pathological cases- incorrect proof structure can reduce to correct ones, ie. it can reduce to a non disruptive structure. This points out that this results is not a characterization of correctness, it even show that correctness cannot be characterized by the cut–elimination methods used in this work (modularity and instantiation).

[46] There is a slight abuse or ambiguity in the use of the word normalizing, it could be that after cut–elimination the structure is a deadlock, fortunately the deadlock is a disruptive proof structure by definition, but also an incorrect one.

So we merely have the inclusion of the class of proof structures normalizing to disruptive proof structure in the class of incorrect proof structure. Therefore we showed equivalently, that the class of proof nets, is smaller than the class of structure normalizing into a non disruptive structure. For this reason Béchet chooses to say that the correctness criterion for MLL is *minimal*.

Such that we would say that a correctness criterion for a certain fragment of LL is *minimal* if the given criterion recognizes[47] only proof structures that are well behaved[48].

[47] We would say that a correctness criterion recognizes a proof structure S, if S is correct according to this criterion.

[48] Also this notion varies from fragment to fragment, but the well behavedness is intended to be relative to the process of cut–elimination, ie. the computational content of proofs.

10 The difficulty of a generalization to MELL

We want here to discuss – briefly – the possibility of a generalization of the Béchet's theorem to proof structures of **MELL**, to do so we must first define these proof structures. The choice of which definition to choose is of importance since, it directly affects the correctness criterion, therefore our goal is to choose the definition of MELL proof structures such that these structures enjoys a correctness criterion as much as possible close to the Danos Regnier criterion. Althought here there are several ways to process, the most common definition seems to be the one that can be find in [Lau01], it enjoys a criterion similar to DR-correctness, athough the notion of switch is a bit changing while one could want a notion of switch that allows us define switching cycles as switching – cyclic – path as is defined in [Ngu20].

There is one definition that could seem to bring a more similar notion of switching, and so a similar criterion to the one of DR-correctness of MLL, it can be seen in [GM01]. The point is to not have $?w$ weakening nodes, but to combine them with axioms. In the end with our previous vocabulary in the graph theory framework, it consists of only having one type of introduction node. The point is that the switching are really similar to the one of MLL.

To generalize the argument we have to adress – at least – these general problems;

- How a confrontation of a MELL-tree without cuts $\mathcal{T}$ against its dual $\mathcal{T}^{\perp}$ resolves after cut elimination in MELL.
- Does the behavior of a MELL proof structure is equivalent to the one of (any of) its $\parr$ closures.
- Can we generalize the notions of configuration and trackers to MELL proof structures, and if so does minimal cyclicity ensures acyclicity in the dual graph.

$$A, B = X \in \mathcal{V} \quad | \quad A^{\perp} \quad | \quad A \otimes B \quad | \quad A \parr B \quad | \quad ?A \quad | \quad !A$$

$$(?A)^{\perp} = !(A^{\perp}) \qquad (!A)^{\perp} = ?(A^{\perp})$$

$$\frac{\vdash \Gamma}{\vdash \Gamma, ?A}\,{}^{?\mathrm{w}} \qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\,{}^{?\mathrm{c}} \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\,{}^{?\mathrm{d}} \qquad \frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\,{}^{!}$$

Figure 32: Formulas, De Morgan laws, and rules of the **MELL** fragment

For MELL proof nets we introduce 4 types of new nodes (ie. labels) to current one of MLL;

- A node label by $?w$ is called a *weakening node* it has no input, and 1 input, which is an arrow labeled by a formula $?A$.

- A node labeled by $?d$ is called a *derelicition* node it has one input and one output, such that that if the input is labeled A the output is labeled by $?A$.
- A node labeled by $?c$ is called a *contraction* node it has two output of the same label $?A$, and one output labeled $?A$.
- Nodes labeled by ! is called a *promotion* node, it has one output labeled by A and an output labeled by $!A$.

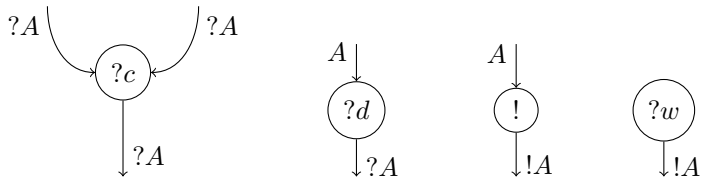


Figure 33: The MELL link (without re-mention the one of MLL).

On the promotion node. Unlike MLL the use of a node in a splitting[49] or internal way, is not enough to render of the correctness of MELL proof structures. The promotion rule needs a specific context, it only is valid if all the conclusions of a sequent $\vdash ?\Gamma, A$ but one are of the form $?B$, then only we can promote the formula A to $!A$, ie. derivate the sequent $\vdash ?\Gamma, !A$. In proof structure this control of the use of the promotion, is done by the notion of boxes.

Definition 10.1 (Box in MELL)**.** We call a box any MELL proof structure B with $n+1$ conclusions, such that at least n of its conclusions are formulaes of the form $?B$. The n "why not" conclusions are called auxiliary doors, while the remaining output labeled by A is called the terminal door.

Then a proof structure S has a correct use of the promotion ! node, if to each promotion node of S we can associate a box B that is a substructure of S and such that the terminal door of B is linked to the promotion node.

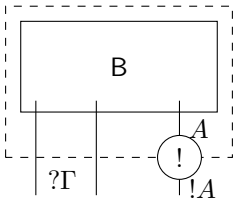


Figure 34: A MELL box associated to a promotion node (the orientation of arrows is top to bottom).

[49]The removal of the node results in a disconnected graph.

MELL–switching. Given a MELL proof structure we define it's switching as the following;

- For each $\parr$ node we choose one input arrow.
- For each $?c$ onde we choose one input arrow.
- For each ! node u and its associated box $\mathcal{B}(u)$, the substructure $\mathcal{B}(u)$ is deleted remaining only u node with all conclusions of $\mathcal{B}(u)$ being its outputs.

The difficulty, A Universal Opponent. Unfortunately there is an evident difficulty in the case of MELL, this can be seen in the case of the cut elimination of the promotion against the contraction, the exponential $! - box$ is after reduction occuring in one module as opposing the first environment resulting in $?A$ while this same box occurs opposing the second environment as well. This is seen in the figure 35.

In order to reveal bad behavior with cut elimination, this would mean that we need to find a sort of 'universal box' of conclusion $!A$, that against any opponent of conclusion $?A$ reduces to a deadlock[50] or a disconnected graph. This tasks seems difficult, so the question of a generalization to MELL remains open.

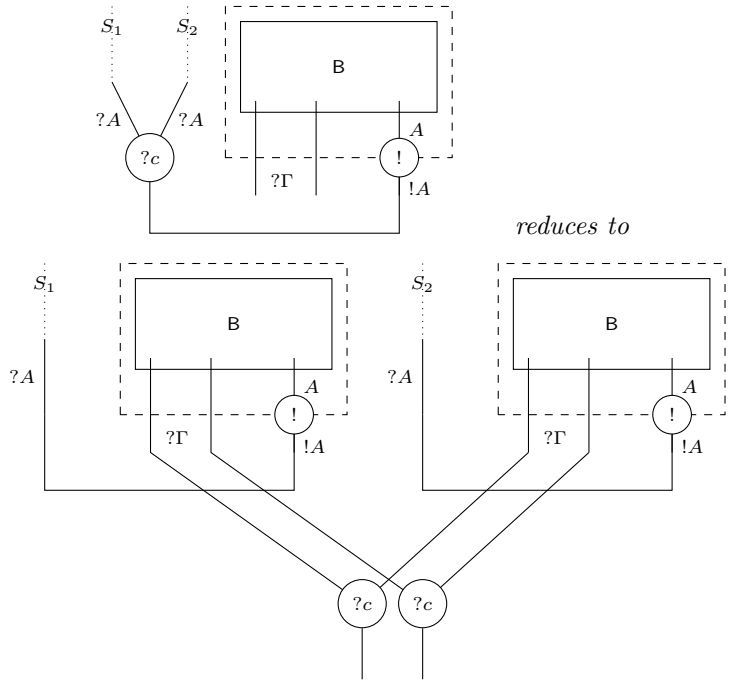


Figure 35: The redex for a promotion against a contraction in MELL proofnets. (as found in [Lau01]).

[50] More precisely it would reduce to a module containing a deadlock.

ACKNOWLEDGEMENT

I first thank Lorenzo Tortora de Falco for supervising this thesis, and giving the original direction of this work. Also, for the freedom he gave me in the writing of this document. And, I'd like to salute his investement in the "Curriculum Binazionale di Laurea Magistrale in Logica", this year has been really enriching for me.

Bibliography

[Abr94] Samson Abramsky. "Proofs as processes". In: *Theoretical Computer Science* 135.1 (1994), pp. 5–9. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(94)00103-0. URL: http://www.sciencedirect.com/science/article/pii/0304397594001030.

[BEC98] DENIS BECHET. "Minimality of the correctness criterion for multiplicative proof nets". In: *Mathematical Structures in Computer Science* 8.6 (1998), pp. 543–558. DOI: 10.1017/S096012959800262X.

[Bef14] Emmanuel Beffara. "A logical view on scheduling in concurrency". In: *Proceedings Fifth International Workshop on Classical Logic and Computation, CL&C 2014, Vienna, Austria, July 13, 2014*. Ed. by Paulo Oliva. Vol. 164. EPTCS. 2014, pp. 78–92. DOI: 10.4204/EPTCS.164.6. URL: https://doi.org/10.4204/EPTCS.164.6.

[Bri08] Michel Brion. "Representations of quivers". In: *Lecture Notes of Summer School "Geometric Methods in Representation Theory"*. Grenoble, France, 2008.

[BS94] G. Bellin and P.J. Scott. "On the -calculus and linear logic". In: *Theoretical Computer Science* 135.1 (1994), pp. 11–65. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(94)00104-9. URL: http://www.sciencedirect.com/science/article/pii/0304397594001049.

[Cur05] Pierre-Louis Curien. "Introduction to linear logic and ludics, part II". In: *CoRR* abs/cs/0501039 (2005). arXiv: cs/0501039. URL: http://arxiv.org/abs/cs/0501039.

[DR89] Vincent Danos and Laurent Regnier. "The structure of multiplicatives". In: *Archive for Mathematical Logic* 28.3 (Oct. 1989), pp. 181–203. ISSN: 1432-0665. DOI: 10.1007/BF01622878. URL: https://doi.org/10.1007/BF01622878.

[Ehr16] Thomas Ehrhard. "An introduction to Differential Linear Logic: proof-nets, models and antiderivatives". In: *CoRR* abs/1606.01642 (2016). arXiv: 1606.01642. URL: http://arxiv.org/abs/1606.01642.

[Gir02] Jean-Yves Girard. "Ludics : An Introduction". In: *Proof and System-Reliability*. Ed. by Helmut Schwichtenberg and Ralf Steinbrüggen. Dordrecht: Springer Netherlands, 2002, pp. 167–211. ISBN: 978-94-010-0413-8. DOI: 10.1007/978-94-010-0413-8_6. URL: https://doi.org/10.1007/978-94-010-0413-8_6.

[Gir87] Jean-Yves Girard. "Linear logic". In: *Theoretical Computer Science* 50.1 (1987), pp. 1–101. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(87)90045-4. URL: http://www.sciencedirect.com/science/article/pii/0304397587900454.

[Gir95] J.-Y. Girard. "Linear Logic: its syntax and semantics". In: *Advances in Linear Logic*. Ed. by Jean-Yves Girard, Yves Lafont, and LaurentEditors Regnier. London Mathematical Society Lecture Note Series. Cambridge University Press, 1995, pp. 1–42. DOI: 10.1017/CBO9780511629150.002.

[Gir96] Jean-yves Girard. "Proof-nets: The parallel syntax for proof-theory". In: *Logic and Algebra*. Marcel Dekker, 1996, pp. 97–124.

[GM01] Stefano Guerrini and Andrea Masini. "Parsing MELL proof nets". In: *Theoretical Computer Science* 254.1 (2001), pp. 317–335. ISSN: 0304-3975. DOI: https://doi.org/10.1016/S0304-3975(99)00299-6. URL: http://www.sciencedirect.com/science/article/pii/S0304397599002996.

[GTL89] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and Types*. USA: Cambridge University Press, 1989. ISBN: 0521371813.

[HP96] Edward Hermann HAUESLER and Luiz Carlos PEREIRA. "GENTZEN'S SECOND CONSISTENCY PROOF AND STRONG CUT-ELIMINATION". In: *Logique et Analyse* 39.153/154 (1996), pp. 95–111. ISSN: 00245836, 22955836. URL: http://www.jstor.org/stable/44084559.

[Kri02] Jean-Louis Krivine. "Lambda-calculus types and models". DEA. Lecture. Université Paris 7, Feb. 2002. URL: https://cel.archives-ouvertes.fr/cel-00574575.

[Lau01] Olivier Laurent. "Etude de la polarisation en logique". 2002AIX22006. PhD thesis. 2001, 230 f. URL: http://www.theses.fr/2002AIX22006/document.

[Mél09] Baptiste Mélès. "La Négation en théorie de la démonstration, de Gerhard Gentzen à Jean-Yves Girard". In: *Séminaire du centre de recherches Philosophies et rationalités*. Clermont-Ferrand, France, May 2009. URL: https://hal.archives-ouvertes.fr/hal-01225141.

[Ngu20] Lê Thành Dng Nguyên. "Unique perfect matchings, forbidden transitions and proof nets for linear logic with Mix". In: *Logical Methods in Computer Science* Volume 16, Issue 1 (Feb. 2020). DOI: 10.23638/LMCS-16(1:27)2020. URL: https://lmcs.episciences.org/6172.

[Oka98] Mitsuhiro Okada. "An Introduction to Linear Logic: Expressiveness and Phase Semantics". In: *Theories of Types and Proofs*. Ed. by Masako Takahashi, Mitsuhiro Okada, and Mariangiola Dezani-Ciancaglini. Vol. Volume 2. MSJ Memoirs. Tokyo, Japan: The Mathematical Society of Japan, 1998, pp. 255–295. DOI: 10.2969/msjmemoirs/00201C070. URL: https://doi.org/10.2969/msjmemoirs/00201C070.

[Par92] Michel Parigot. "$\lambda\mu$-Calculus: An algorithmic interpretation of classical natural deduction". In: *Logic Programming and Automated Reasoning*. Ed. by Andrei Voronkov. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 190–201. ISBN: 978-3-540-47279-7.

[Pra65] Dag Prawitz. *Natural Deduction: A Proof-Theoretical Study*. Dover Publications, 1965.

[PT10] Michele Pagani and Lorenzo Tortora de Falco. "Strong normalization property for second order linear logic". In: *Theoretical Computer Science* 411.2 (2010), pp. 410–444. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2009.07.053. URL: http://www.sciencedirect.com/science/article/pii/S0304397509005180.

[Ret96] Christian Retoré. "Perfect matchings and series-parallel graphs: multiplicatives proof nets as RB-graphs: [Extended Abstract]". In: *Electronic Notes in Theoretical Computer Science* 3 (1996). Linear Logic 96 Tokyo Meeting, pp. 167–182. ISSN: 1571-0661. DOI: https://doi.org/10.1016/S1571-0661(05)80416-5. URL: http://www.sciencedirect.com/science/article/pii/S1571066105804165.

[SU10] Morten Sørensen and Pawe Urzyczyn. "Lectures on the Curry-Howard Isomorphism". In: *Studies in Logic and the Foundations of Mathematics* 149 (Oct. 2010). DOI: 10.1016/S0049-237X(06)80005-4.

[Tra11] Paolo Tranquilli. "Intuitionistic differential nets and lambda-calculus". In: *Theoretical Computer Science* 412.20 (2011). Girards Festschrift, pp. 1979–1997. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2010.12.022. URL: http://www.sciencedirect.com/science/article/pii/S0304397510007139.

[UB99] Christian Urban and Gavin Bierman. "Strong Normalisation of Cut-Elimination in Classical Logic". In: vol. 45. Jan. 1999, pp. 645–645. DOI: 10.1007/3-540-48959-2_26.

List of Figures

A Duality, minimal cyclicity–minimal acyclicity

In this annexe we try to explicit an equivalency between minimal cyclicity and a kind of acyclicity we will say to be minimal.

Minimal Acyclicity. From minimal cyclicity comes its counterpart that is *minimal acyclicity*. Althought this notion is way less intuitive, and in fact it has been thought of in the demonstration of the lemma 9.1. Here we give no interesting meaning to minimal acyclicity, merely defining it as the dual notion of minimal cyclicity, but maybe it could be interesting to question what minimal acyclicity corresponds to for proof structures (and proofs). If we see minimal cyclicity as the minimal way to fail the DR-correctness criterion by having a swicthing cycle – or even by considering the fragment **MLL+Mix** – then what does it mean to successfully pass the test of switching cycle, and so be DR-correct in a minimal way? In that way we could distinguish two types of correct structures – and so proof nets –, acyclic minimal proof nets, and non acyclic minimal ones, would that carry a relevant meaning in any way?

Namely we say that a configuration $\mathcal{C}$ is *minimaly acyclic*, if

- $\mathcal{C}$ is acyclic.
- Each pairs of $\mathcal{C}$ meet on distinct $\parr$ nodes in $\mathcal{T}$.
- There exists some tracker T without invariant, such that the transposed configuration $T(\mathcal{C})$ does not contain any cycle with invariant in the dual module – and so equivalently meeting in distinct parr's in $\mathcal{M}$ – (ie. of length smaller that $n = card(\mathcal{C})$).

It seems minimal acyclicity is relative to a tracker, althought we will show it does not have to, we try to reformulate the 3rd point.

Assume $T(\mathcal{C})$ has a cycle with invariant this would mean the existence of a cylic tracker (μ, π'_1, π'_2), this means each pairs $\{\pi'_2(p), \pi'_1(\mu(p))\}$ meet in distinct $\parr$'s in $\mathcal{M}$. And μ has an invariant. Then it means that the collection of the pairs $\{\{\pi'_2(p), \pi'_1(\nu(p))\} \mid p \in supp(\mu)\}$ is acyclic and each pairs meet in distinct parr's, note that at least one these pairs is not a pair of $\mathcal{C}$.

Note that a pair d of $T(\mathcal{C})$ is of the form $\{\pi_2(p), \pi_1(\sigma(p))\}$ therefore, $\pi'_2(d)$ is either $\pi_2(p)$ or $\pi_1(\sigma(p))$.

Further more, given a pair d of $T(\mathcal{C})$, then $\mu(d) = \{\pi_2(p'), \pi_1(\sigma(p'))\}$ so we define a permutation $\nu : \mathcal{C} \to \mathcal{C}, p \to p'$. Note that since μ as an invariant so does ν by construction.

Then like before, $\pi'_1(\mu(d))$ is also $\pi_2(\nu p)$ or $\pi_1(\sigma(\nu p))$ when $d = \{\pi_2(p), \pi_1(\sigma(p))\}$.

Consider p an invariant of ν, then necessarily $\{\pi'_2(p), \pi'_1(\nu(p))\}$ corresponds to $\{\pi_2(p), \pi_1(\sigma(p))\}$, which is not a pair of $\mathcal{C}$.

Therefore the last condition could reformulated as there exist a configuration $\mathcal{C}'$ of cardinal lower than the one of $\mathcal{C}$, such that $\neg(\mathcal{C}' \subset \mathcal{C})$ is acyclic and each pairs meet in distinct parr's.

If such a thing is true, then let's construct a tracker in $T(\mathcal{C})$ of a cycle. Consider the pairs of $T(\mathcal{C})$ that are $\mathcal{P} = \{\{\pi_2(p), \pi_1(\sigma(p))\} \mid \bigcup p \in \bigcup \mathcal{C}'\}$.

Then we merely define a permutation μ as taking one pair $p' \in \mathcal{P}$, then $p' = \{\pi_2(p), \pi_1(\sigma(p))\}$ note that $\pi_2(p) \in \bigcup \mathcal{C}'$ so there exists a pair $\{\pi_2(p), \beta\}$ in $\mathcal{C}'$. Define $\pi'_2(p') = \pi_2(p)$.

Then β is in $\bigcup \mathcal{C}'$ as well so there is a pair $\{\beta, \gamma\}$ in $\mathcal{P}$. The application $\mu : p' \to \{\beta, \gamma\}$ is a permutation. We define $\pi'_1(\mu(p')) = \beta$.

By hypothesis on $\mathcal{C}'$; $(\alpha, \beta) = (\pi_2(p'), \pi'_1(\mu(p')))$ meet in nodes in the dual and all meeting points are distinct, this means we track a cycle. And the tracker (μ, π'_1, π'_2) has an invariant which goes against minimal acyclicity.

We could question the meaning of minimal acyclicity by trying to provide an answer to the following questions;

- What does it mean for a proof π of MLL, that in its associated proof net $[\pi]$, an axiom node meet in a $⅋$ node ?
- Given a proof net $[\pi]$, what does it mean for π, that all axiom nodes meet in distinct points ?
- What finally does it mean that we can find a smaller configuration for the proof net $[\pi]$, such that the proof structure (it would not be a proof net since they are less axioms) $\mathcal{C}(\mathcal{B}_{[\pi]})$ have all its axiom nodes meeting in $⅋$ nodes?

Lemma A.1 (Duality cyclicity–acyclity.)**.** *Given a module $\mathcal{M}$ of labels $\otimes$ and $⅋$ and $\mathcal{C}$ a configuration of $\mathcal{M}$, the two statement are equivalent;*

1. *$(\mathcal{M}, \mathcal{C}, T)$ is minimaly cyclic.*
2. *$(\mathcal{M}^\perp, T^\perp, T(\mathcal{C}))$ is minimaly acyclic.*

Proof. **(1⇒2).** Given the configuration $\mathcal{C}$ and its tracker T, let's us show that the transposition $T(\mathcal{C})$ generates no switching cycles.

Acyclicity. Ad absurdum, assume that $T(\mathcal{C})$ generates a cycle of length $m \leq n$. Consider therefore the tracker (μ, π'_1, π'_2) of this cycle.

Consider $d_1, ..., d_m$ an enumeration of the support of μ. Then consider the collection of arrows, $\mathcal{F} = \{\{\pi'_1(d_k), \pi'_2(d_k)\} \mid 1 \leq k \leq m\}$. Then let's explicit two collections of meeting points;

- We know that each pair $(\pi'_1(d_k), \pi'_2(d_k))$ meet in distinct $⅋$'s in $\mathcal{M}^\perp$ ensuring m meeting points. This is because $d_k = \{\pi_2(p), \pi_1(\sigma(p))\}$ and by assumption on the tracker T to be cyclic, we know that these pairs of arrows meet in distinct $\otimes$ in $\mathcal{M}$.
- On the other hand, assuming the tracker (μ, π'_1, π'_2) describes a cycle means by definition $(\pi'_2(d_k), \pi'_1(\mu(d_k)))$ meet in distincts $\otimes$ nodes, ensuring m meeting points labeled by $\otimes$ in $\mathcal{M}^\perp$.

But note that none of these two collections of m nodes cannot overlap, since one corresponds to nodes labeled by $⅋$'s while the other one corresponds to nodes labeled by $\otimes$ (in $\mathcal{M}^\perp$). This ensure that the family $\mathcal{F}$ has $2m$ meeting points. But this goes against the upper bound of meeting

points, the proposition 4.10 ensures that there is at most $2m-1$ meeting points for a family of $2m$ arrows. We have a contradiction, therefore the transposition $T(\mathcal{C})$ does not generate any cycle of length m.

Minimality. Taking the dual tracker $T^{\perp}$ we can ensure from the previous proposition that $T^{\perp}(T(\mathcal{C})) = \mathcal{C}$, and indeed $\mathcal{C}$ does not have any cycle length $m < n$ – by assumption – in $\mathcal{M}$.

Also each pairs $\{\pi_2(p), \pi_1(\sigma(p))\}$ meet in distinct $\parr$ nodes in $\mathcal{M}^{\perp}$, since T describes a cycle for $(\mathcal{C}, \mathcal{M})$.

T having no invariant (since it tracks a minimal cycle), it naturally follows from the definition of $T^{\perp}$, that it also does not have invariant.

This show that $T(\mathcal{C})$ is minimaly acyclic configuration of $\mathcal{M}^{\perp}$ (for the tracker $T^{\perp}$).

(2⇒1). Note that by renaming $\mathcal{C} := T(\mathcal{C}), T := T^{\perp}$ and $\mathcal{M} := \mathcal{M}^{\perp}$ we can merely focus on showing the following ; $\mathcal{C}$ minimaly acyclic in $\mathcal{M}$ implies $T(\mathcal{C})$ minimaly cyclic in $\mathcal{M}^{\perp}$.

No subcycles. Since $\mathcal{C}$ is minimaly acyclic for T, this by definition means $T(\mathcal{C})$ does not generate any cycles of length $m < n$.

Full Cycle. Let's show that $T(\mathcal{C})$ induces a cycle of length n ie. without invariant. Recall $T(\mathcal{C})$ corresponds to the pairings $\{\pi_2(p), \pi_1(\sigma(p))\}$ where $p \in \mathcal{C}$. Minimal acyclicity of $\mathcal{C}$ ensures that each $\pi_1(p), \pi_2(p)$ meet on distinct $\parr$ nodes in $\mathcal{M}$ – and so in distinct $\otimes$ node in $\mathcal{M}^{\perp}$ –.

Consider a pair of $T(\mathcal{C})$ that is $p' = \{\pi_2(p), \pi_1(\sigma(p))\}$ and recall that $\sigma^{\perp}(p') = \{\pi_2(\sigma(p), \pi_1(\sigma\sigma(p))\}$. Note that therefore the minimal acyclicity, ensures that $\pi_2^{\perp}(p') = \pi_1(\sigma(p))$ and $\pi_1^{\perp}(\sigma^{\perp}(p')) = \pi_2(\sigma(p))$ meet in distinct $\otimes$ nodes in $\mathcal{M}^{\perp}$.

This exactly means that $T^{\perp}$ tracks a cycle for the configuration $T(\mathcal{C})$ in $\mathcal{M}^{\perp}$. Therefore, meaning that $T(\mathcal{C})$ is a minimaly cyclic configuration of $\mathcal{M}^{\perp}$.

□

B Detailed Abstract

Correctness as Good Interactive Behavior

Detailed Abstract : introduction

The base of this thesis is an article written by Denis Béchet in 1998 and titled, "Minimality of the correctness criterion for multiplicative proof nets" see [BEC98]. We rework the article with more modern notions to describe the geometry of proof–nets but also the process of cut–elimination.

Proof–nets is a new representation of proofs that appeared with Girard linear logic, introduced in 1987 [Gir87]. In this representation proofs corresponds to graph[51] while the cut–elimination correspond to graph rewriting. The thing is that proof–nets belong to a more general class of graphs that are called proof–structures, and not all proof structures are proof–nets[52]. For this reason there has been a search for correctness criterion that characterize proof structures that are proof nets. The most famous for the MLL[53] fragment of linear logic is due to Danos and Regnier [DR89]. This criterion is a rather geometrical one based on a notion of switching[54], Béchet wondered in [BEC98] what this criterion would mean for cut–elimination. Béchet succesfully showed that DR[55]–correctness corresponds to a notion of good behavior relatively to cut–elimination for cut free proofs. The notion of *good behavior* is defined in the paper of Béchet, negatively, meaning he introduces a notion a *bad behavior* and then simply define good behavior as its negation. Proof–structures that behave badly are the one that interacting[56] with a correct proof–structure[57] reduce after cut elimination to a disconnected graph or a *deadlock*. A deadlock is an important proof structure that corresponds to an axiom node above a cut node.

Now we try to shed some light on what can motivate such a work. First why linear logic? Linear Logic can be seen as a refinement of classical logic, although discovered through denotational semantic of the λ–calculus, perhaps it's most important feature is the control of the rule of weakening and contraction of classical logic. The rules of weakening are indeed natural to consider in classical logic, the intuition in classical logic is that a sequent of the $\Gamma \vdash \Delta$ is intended to mean $\bigwedge \Gamma \to \bigvee \Delta$.

This intuition is no longer true in linear logic, where sequents are understood as $\bigwedge \Gamma \multimap \bigvee \Delta$[58]. The difference is born in what the linear implication $\multimap$ really

[51] One can find the expression proofs–as–graphs.
[52] Meaning they represent a proof.
[53] This stands for Multiplicative Linear Logic.
[54] A transformation of the graph that may create disconnection, or leave cycles.
[55] This stands for Danos–Regnier.
[56] Here it means we link conclusions of the two graphs by a cut node.
[57] And so by characterization a proof–net.
[58] Here there is a slight abuse since in linear logic the connectors $\vee$and $\wedge$do not *really* occur, instead they are both decomposed in two connectors

is, namely, the transformation – or consumption – of a formula into another. [59]

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\, w-R \qquad \textit{and} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\, w-L$$

$$\frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\, c-R \qquad \textit{and} \qquad \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\, c-L$$

Linear logic is also following (one of) the intention of Gentzen sequent calculus for classical logic [Mél09]. What do we mean by this; the sequent calculus LK manipulates sequent of the form $\Gamma \vdash \Delta$ and the classical negation is generating a strong symmetry. This is expressed in the rule $(\neg_L - intro)$ and $(\neg_R - intro)$ of LK, these rules states that any formula A can pass on the other side of the sequent $\vdash$ by being negated, this also allows for one sided sequent in LK. This symmetry is just not present in intuistionistic logic.

$$\frac{\Gamma \vdash \Delta, A}{\Gamma, \neg A \vdash \Delta}\, \neg_L - intro \quad \textit{and} \quad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\, \neg_R - intro$$

Linear logic pursue this same direction, towards more symmetry. In fact through the control of weakening and contraction, it becomes apparent that there is a need to split the classic connectors $\wedge$ *"and"* and $\vee$ *"or"* in two versions, an additive and multiplicative one. Not doing so would not lead to a full control of the contraction and weakening there is a classic exemple with the use of the *"and"* connector[60]. In LL then, we have four connector $\otimes, \&$ the multiplicative and additive *"and"* and $⅋, \oplus$ the multiplicative and additive *"or"*. The symmetry again exhibited by the negation, it is that $\otimes$ and $⅋$ are duals, while $\&$ and $\oplus$ are duals. The idea is that negation does not change the multiplicative or additive character of a connector. This symmetry goes even further, also working with the units.

$$\frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\, \otimes \qquad \frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \,\&\, B}\, \&$$

What can motivate the proofs–as–graphs principle; with linear logic the new geometrical representation of proofs as a direct interest, it does not involve the unnecessary *bureaucraty* (an expression that comes from Girard) that can occur in the tree representation, for exemple the order on which we use a $⅋$ as conclusion. To reduce the bureaucracy is also a direction that one can go towards, since it means that we have less proofs to study : two proofs of the same sequent π_1, π_2 may have the same graphical representation, meaning that they only differ in bureaucratic aspects, not in logical ones. Indeed for proof theory this is an important feature to try to get a better understanding of what is *really* the logical content of a proof.

[59] But still, the *classical implication* can be decomposed in linear logic as $A \to B = !A \multimap B$, using the linear implication $\multimap$ and the exponent !. To give an intuition, the exponentiation $!A$ can be understood as "there are infinitely many A's" or "A is an eternal truth".

[60] When the context Γ and Δ are the same.

$$\dfrac{\dfrac{\vdash \Gamma, A, B, C}{\vdash \Gamma, A \parr B, C}\parr}{\vdash \Gamma, (A \parr B) \parr C}\parr \quad \textit{is really similar to} \quad \dfrac{\dfrac{\vdash \Gamma, A, B, C}{\vdash \Gamma, A, B \parr C}\parr}{\vdash \Gamma, A \parr (B \parr C)}\parr$$

This representation also comes with a new object, the *proof structures*. They are a class of graphs of which some represents proofs of LL, which if the case occurs they are called proof nets. This apparent problem leads to a new question that is the search of correctness criterion to characterize proof structures that are proof nets. This new aspects draws stronger link between logic and graph theory, sure with Gentzen's work proofs where seen as trees, but the link with graph theory was still limited to this small class of graphs that are trees. The proof structures are a way more general class of graph, and in fact the correctness criterion has been shown to be related to a classic problem of graph theory that is the notion of *perfect matching*[61], see [Ngu20] and [Ret96].

Now we would like to draw the links between computer science and this work, this is – of course – based on cut elimination. One the strongest link between computer science and more specifically functional programming and proof theory is the Curry–Howard correspondence : the correspondence also known as the proofs–as–programs paradigm, states that proofs of Nm[62] corresponds to terms of the simply typed lambda calculus, while the formulas correspond to types and the cut elimination (or detour elimination for natural deduction) corresponds to the β–reduction, ie. the execution of programs.

$$\dfrac{\vdash \Gamma, A \quad \vdash \Delta, \neg A}{\vdash \Gamma, \Delta}\, cut$$

Since the discovery of this correspondence, logicians and computer scientist have tried to give a computational meaning to cut elimination with different deductive and typing systems. A successfull exemple is the correspondence that occurs with the system F and the natural deduction of the second order Intuistionistic logic, it has been shown by Girard and a formulation of this correspondence can be found in Proofs and Types [GTL89]. There was also attempt for the classical logic with Parigot's $\lambda\mu$–calculus in [Par92]. But note that it is still not always clear what cut elimination corresponds to computationally.

Among other things Linear Logic comes with a built–in notion of temporality that is not found in classical logic. This temporality is also corresponding to a conception of formulas as ressources, and the linear implication as a transformation of the ressource A into the ressource B. In fact there is a correspondance of differential proof nets and a form of enriched lambda calculus that is ressource sensitive, this types of calculus may be called ressource calculus. That kind of work can be found in [Tra11] or [Ehr16].

[61] A perfect matching of a graph corresponds to a set of edges without common vertices that matches each vertex of the graph, meaning each vertex is in the border of an edge.

[62] This denotes the so called Minimal logic, limited to the connector of implication $\to$ and eventually the connector "and" $\wedge$.

But with linear logic new ideas have emerged regarding the correspondence and the computational meaning of cut–elimination. Abramsky suggested that linear logic could be related to process calculus, and specifically the π–calculus, in what he presented the proofs–as–processes paradigm [Abr94], it has later been explore by others to name a few, Bellin and Scott in [BS94] and Beffara in [Bef14]. The great feature of this new correspondence, is that unlike for exemple the $\lambda\mu$–calculus of Parigot, it extend the scopes of the correspondence, no more limiting it to functional programming, and so giving hope for a stronger link between proof theory and computer science (in this case to concurrency).

In the paper [BS94] cut elimination is correponding to a notion information flow and to sending/receiving protocols. Proof–structures are also involved in this new correspondence, and so is the theory of slicing that Girard introduced in [Gir87],[Gir96], it is related to MALL the multiplicative and additive fragment of linear logic. Such a correspondence would also be of interest in concurrency, since many processes calculus exists and such a correspondance could exhibit a candidate among the available calculus to keep, it would also bring the formal tools of proof theory to study problems of computer science.

The study of cut–elimination in LL proof–nets could therefore be a key ingredient towards a better understanding of the possible correspondence, but also eventually of results on concurrency or functional programming.

The work of Béchet is therefore in the line of what has been exposed with the Curry–Howard correspondence, involving the new notions that came with linear logic and the Geometry of interaction. Taking the new geometrical aspects of the correctness criterion for proof structures and questionning it relatively to cut elimination – that is the computational processes of proofs – this work aims towards getting a better understanding of the computational content of proofs, and doing so, of the new object that are the proof–structures.

The thesis is organized in the following way.

- The first three chapters are introductions to key notions for the understanding of the paper. We do not go as much as it would be needed into the details but try to record some important elements.
  - The section 1 is thought of as an incomplete introduction to proof theory, starting from the natural deduction of gentzen, we expose the notion of derivation as trees. We try to point out some differences between intuistionistic and classical logic, and expose the rules of inference of these deductive system (in natural deduction). We record the theorem of cut elimination for the sequent calculus of classical logic.
  - The section 2 is less related to the thesis but is here to illustrate the relation between the cut elimination and the β–reduction, ie. the executions of programs. We present the untyped λ–calculus and expose some classical results. We then present the simply typed lambda calculus as it can be seen in proofs and types (using the

type generators $\to$ and $\times$). And we point out the relation between the natural deduction for intuistionistic logic and the simply typed lambda calculus, the well-known Curry–Howard correspondence.

  - The section 3 is a small presentation of linear logic, we point using traditionals exemple the differences between classical and linear logic, among others a 'built–in' notion of temporality. We then expose the multiplicative fragment of linear logic denoted MLL, and the notion of MLL–proof–structures. We expose the need of correctness criterion, and present the one of Danos–Regnier (which is crucial to get into the article of D. Bechet [BEC98]). We also present the cut elimination, and point the structure that is the deadlock, which will also be of importance for us.

- The fourth section is still kind of introductory, it consists of notions of graph theory in order to describe more precisely the proof structures – and the modules – but, we aim to proove a theorem, which we choose to call the *Bechet's Cyclicity Criterion* a characterization of cycles in some types of graphs. We also present an upper bound in the number of successive meeting point of a family of arrows, which is a key element for a later proof of acyclity.

  - In the first subsection we expose elementary notions of graph theory, but more precisely of the theory of quivers, this object is used mainly in algebra. We define a notion of path in a quiver, and give a proposition to decompose path as composition – ie. correct enough concatenation – of descending and ascending path.
  - In this second subsection, we point out that a cycle can always be described in a standard way, meaning in a certain decomposition that we call standard.
  - In the third subsection, we define the notion of meeting points, in some specific graphs. Then we give an upper bound on the meeting points of a family of arrows with the right condition (compatible with modules) on the quiver. Finally we expose the cyclicity criterion in terms of meeting points. Then we give a formal definition of the modules using the exposed quiver theory, and call a definition presented in a paper of Nguyên [Ngu20], that is well compatible with the cyclicity criterion exposed.

- The fifth section is a formalisation of the notion of module (mainly for MLL), using the formalism of quivers presented in the previous part. We also define the notion of switching.

  - The first subsection is dedicated to giving the definition of modules, switching and proof structures using the previous quiver formalism. We also record the Danos-Regnier correctness criterion, althought we give no proof for it here, we invite the reader to have a look at [DR89] for a proof of the criterion.

  - The second subsection is dedicated to proving a theorem of completion. We seek to find correctness criterion for modules, in order to identify the modules that correspond to sub–modules of proof–nets. Such a criterion can be found, we choose to call it the DR^- correctness criterion and seek to show that it characterize the submodules of proof–nets. This argument can be found in the original work of Béchet [BEC98], but we differ slightly on the presentation of the proofs, trying to make it more readable.

- The sixth section presents the notion of *bad behavior* as Bechet defined it in the original article [BEC98], then we present the ∇–closure and we show there that a proof behaves just like any of its ∇–closure, this is a commodity to study behaviors since then we can focus on only treating trees, instead of modules.

- The seventh section is a look at cut elimination for modules;
  - In its first subsection, we show that a definition of cut elimination using the notions of graph theory from the section 3, although realizable, ends up being too heavy.
  - In the second subsection, we present a (partial) solution that can be found in the paper of Curien [Cur05], the paper is an introduction to ludics, but first exposes some notions to get into ludics. In this framework module are seen as forest of formulaes representation, axioms corresponds to disjoint pairs of the occurency, while the cuts are disjoint pairs of relatives adresses. We can this way describe modules as expressions – but they may become heavy – and so cut elimination as rewriting.

- The eighth section treats the case of a proof structure with a disconnected switching σS, showing that with the right opponent the structure reduces to its switching σS.

- The ninth section treats the case of a proof structure with a switching cycle
  - In the first subsection, we expose the *bechet transformation*, that transform a module containing a minimal switching cycle into an acyclic module. But the transformation is not taking into account the types of the arrows, it is in the end more of geometrical argument and it is based on elements of the section 4.
  - In the second subsection we show that any acyclic configuration (ways to place the axiom) can be made into a well typed one. The argument is rather simple, it is based on the instantiation ϕ introduced by Bechet in the original article.
  - In the third subsection, we show that the confrontation with the created opponent will indeed reduce to a deadlock. To do so we use the notions of "pre-ludics" that we exposed in section 6.

- The tenth section briefly discuss of the possibility of a generalization to the MELL fragment, although it is rather incomplete, we want to point out a difficulty that could arise with contraction.

- The appendix A tries to generalize the lemme 9.1, that state that a minimal cyclic configuration can become an acyclic one. In fact we introduce a notion of minimal acyclicity and show that it is the dual notion of minimal cyclicity. We do not give an interesting meaning to minimal acyclicity, but exhibit some questions that can come with such a notion.

Detailed Abstract : Main Results

The first key result is an identification of paths in a quiver $G = (V, Arr, \mathsf{src}, \mathsf{tgt})$ as the concatenation of ascending and descending paths.

Proposition B.1 (Decomposition of paths)**.** *Given a path ρ, ρ corresponds to a composition of ascending and descending paths. Meaning there exists a number k called the* number of inversion *and a function $D_\rho : \{1, .., k\} \to Arr^*$, such that;*

- $\rho = \prod_{1 \leq i \leq k} D_\rho(i)$.
- *$D_\rho(1)$ is either an ascending or a descending line.*
- *For any index $D_\rho(i)$ is an ascending (resp. descending) line $\Rightarrow$ $D_\rho(i+1)$ is a descending (resp. ascending) line.*

This results is a key step to get to a characterization of cycles, as given in Béchet work [BEC98]. Before getting there we introduce the notion of meeting point, the first point in the intersection of tracks of two path. We also get an important result that is an upper bound on the number of meeting points. This result is important to be able to show a property of acyclicity later.

Proposition B.2 (Meeting points bounding)**.** *Given a binary input quiver, and given $(\rho_1, ..., \rho_n)$ a family of descending paths. Denoting m_i the meeting point of ρ_i, ρ_{i+1}, and m_n the meeting point of ρ_n, ρ_1.*

Then the set of meeting points $\{m_1, ..., m_n\}$ is made of at most $n-1$ nodes.

Finally we get to a characterization of cycles in quivers. This theorem is also important since we will use it to reveal switching[63] cycles in proof structures or modules.

Theorem B.1 (Bechet's Cyclicity Criterion)**.** *Given a quiver G strongly oriented and sequential. The two statements are equivalent;*

1. *There exists n introduction node[64] $N_1, ..., N_n$ and two sequences of arrows of the same length n, $\alpha_1, ..., \alpha_n$ and $\beta_1, ..., \beta_n$ such that*

[63] Cycles in the switched graph.

[64] Meaning nodes with no inputs. That is a node such that it is the target of no arrow.

(a) *(Common Source). For each i, both arrows α_i and β_i are distinct and have their source in N_i.*

(b) *(Meeting arrows). For each i, the two arrows $\alpha_i, \beta_{s(i)}$ meet.*

(c) *(No Redundancy). The meeting points $\alpha_i \wedge \beta_{s(i)}$ are pairwise distinct.*

2. G contains a cycle.

In the next part the theorem of completion also presented in Béchet's work [BEC98] is of importance since, it ensures that any DR^- correct module can be completed in a proof net, therefore, we we look for a correct proof structure we can merely look for a correct module (that is cut free and with one conclusion), this makes the search for proof-net opponents easier.

Theorem B.2 (Completion)**.** *Given a cut–free module $\mathcal{M}$ with one conclusion.*

If $\mathcal{M}$ is DR^- correct, then there exists a cut–free module $\mathcal{M}^*$ with dual entering arrows, such that connecting $\mathcal{M}$ and $\mathcal{M}^*$ by axioms results in a proof net.

From there we have two cases to treat, given a cut free proof structure (with one conclusion), we want to show that if it is incorrect, linking its one conclusion to the one conclusion of a proof net, will reduce to a disconnected graph or a graph containing a deadlock. To do so we therefore treat two cases, one were S has a disconnected switching, and one where it has a switching cycle.

For the case of disconnection, we need two results, one assuring that the opponent that we construct is a proof net.

Proposition B.3 ($\otimes, \parr$-Trees generate proof nets)**.** *Given a tree $\mathcal{T}$ made of $\parr$ and $\otimes$ node. And given σ a switching of $\mathcal{T}$.*

Then the structure $\mathcal{T}^\perp \overset{\ulcorner}{\underset{id}{\urcorner}} \sigma\mathcal{T}$ is a proof net, ie. a correct proof structure.

And a second one about cut elimination.

Proposition B.4. *Given a proof structure S without cuts and made of one conclusion. σ a switching of S. The connecting structure $S \llcorner\lrcorner \mathcal{B}_S^\perp \ulcorner\urcorner \sigma\mathcal{B}_S$ reduces exactly to σS.*

It is clear that then if σS is disconnected, that we have the result that we want. Noting that $\mathcal{B}_S$ is a cut free MLL tree.

Now for the case where S has a switching cycle. We first explicit a transformation that we call Bechet transformation of graphs that are pre–modules, basically MLL modules that are unlabeled. Then using this transformation with MLL modules we reveal that it transform a minimaly cyclic module into an acyclic pre–module.

Lemma B.1 (The transposition of a minimaly cyclic configuration is acyclic in the dual.)**.** *Given a module $\mathcal{M}$ of labels $\otimes$ and $\parr$ and $\mathcal{C}$ a configuration of $\mathcal{M}$, the two statement are equivalent;*

If $(\mathcal{C}, T)$ is a minimaly cyclic configuration of $\mathcal{M}$ then its béchet transformation $T(\mathcal{C})$ is an acyclic configuration in the dual $\mathcal{M}^\perp$.

This result is mainly geometric, the acyclicity is ensured but it is not ensure that that the pre–module obtained after the transformation is indeed a module, therefore we call the instantiation ϕ to ensure that we can create a 'well–typed' module.

Proposition B.5 (Generating a correct module)**.** *Given a module $\mathcal{M}$ and $\mathcal{C}$ an acyclic configuration of $\mathcal{M}$.*

Then $\phi(\mathcal{C})[dev(\phi(\mathcal{M}))]$ is correct module.

Note that with this result, we show that the module is DR^- correct and so here the theorem of completion will be usefull to ensure the completion into a proof–net. Then it remains only to show that the interaction of S with the constructed opponent reduces to a deadlock. In fact we can show that this is true even if S does not have switching cycle, the switching cycle is only of use to ensure the correctness[65] of the opponent.

Proposition B.6 (A Dual confrontation that reduces to deadlocks)**.** *Given $\mathcal{C}$ a configuration of the partition U, a subset $\mathcal{C}_0 \subset \mathcal{C}$ and T some tracker of $\mathcal{C}_0$. Given $\mathcal{T} = [U : \phi(A)]$*

Then the module $\dfrac{\xi\mathcal{C}, \nu T(\mathcal{C}_0)}{[\xi, U : \phi(A)] \times [\nu, U + 2 : \phi(A^{\perp})]}$ *will reduce to a deadlock containing module.*

Finally we can conclude on the equivalency. Also recording that the other direction of the implication is rather easy given that we have knowledge of DR correctness, one can look at [DR89].

Theorem B.3 (Béchet, 1998 [BEC98])**.** *Given a proof structure S without cuts. The two statements are equivalent;*

1. *S is incorrect.*
2. *S is a bad proof structure.*

Here a bad proof structure, is a proof structure that does not behave well in the interaction – ie. cut elimination – with proof nets.

[65] The well typedness and the deadlock reduction can be ensured without assuming the S as a switching cycle, but to ensure that the opponent has no switching cycle we have to use the fact that S as a switching cycle.