Roberto Di Cosmo

roberto@dicosmo.org

http://www.dicosmo.org

Introduction to

LINEAR LOGIC

Notes for the MPRI course

Extended version of the Linear Logic Primer,
originally written in collaboration with Vincent Danos

N.B.: this is unfinished work, made available for your convenience.

Contents

Preface

Linear Logic was built by Jean-Yves Girard around 1986, and was originally motivated by a deeper investigation of the semantics of λ-calculus. Such investigation led to the surprising discovery that, in *coherence spaces*, the intuitionistic implication, or arrow-type $A \Rightarrow B$ is interpreted, as is now very well known, as $!A \multimap B$, i.e. it is decomposed into two *more primitive* constructs, a *linear* implication $\multimap$ and the modality $!$.

Only afterwards the new connectives were understood from the syntactic point of view.

In these notes, though, we will deliberately present Linear Logic in a very partisan way: we will stick to the syntactical viewpoint, and concentrate all our efforts in familiarizing the reader with the Linear Logic system, while referring for the detailed investigation of the formal semantics to one of the nice presentations that are already available (see for example [Gir87, GLT90]).

These notes born in Paris, in 1992, during two weeks of intensive work which allowed the second author to have an exciting tour of Linear Logic, guided by the first one. Some parts are already published and well known, but other ones provide an original presentation of material that can be found mostly in the first author's Phd thesis.

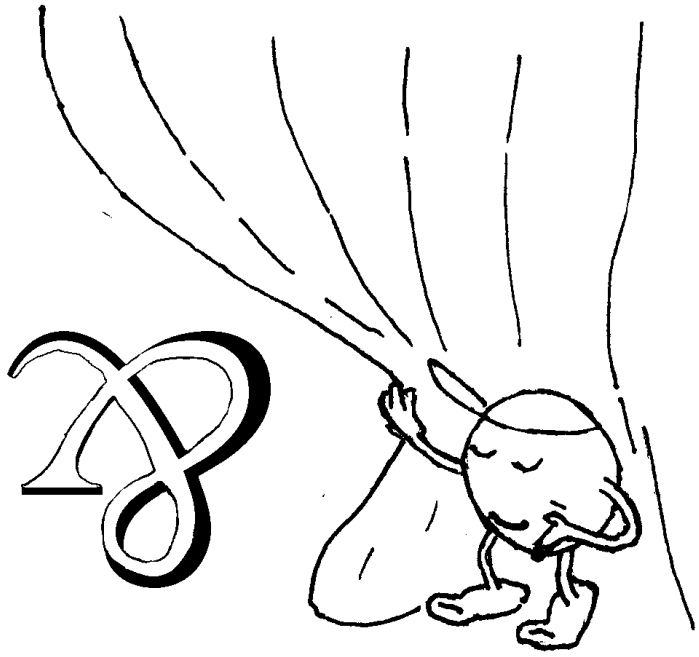
There is a huge amount of exercises: many of them are completely new ones that are especially selected in order to familiarize the reader with some important fine points and are therefore intended as an essential part of the course notes, *not* as a decoration.

Finally, remark that all the presentation given here is inside the world of *propositional* Linear Logic. It is amazing to see how much can be said about propositional calculus, and how much propositional Linear Logic has to say about *computation*, that is here our major concern.

CHAPTER 1

Introduction

I'm not a linear logician
Jean-Yves Girard
Tokio, April 1996



Linear Logic is a transversal theme: nobody really claims to be purely a linear logician, not even its very inventor, but many people have discovered that Linear Logic provides valuable tools ranging from simply the new language, that conveys better intuitions, to new logical, semantic and algebraic structures. On this tenth anniversary of Linear Logic, even a superficial look at the titles of the over 300 papers in the literature is sufficient to confirm this peculiar aspect: only a few papers are about Linear Logic itself, while the vast majority are examples of how these tools have been fruitfully applied in practically all disciplines of theoretical computer science, and in many cases trickled down to real world applications.

It may well be the case that some problem or application you are currently working at could benefit from the linear logic toolbox, and the goal of this book is precisely to help you find out if this is the case. For this, we think that it is important to provide a lightweight introduction to linear logic, and then give a broad view on the elementary concepts that turned out to be the most fruitful in applications. We will systematically detail the elementary cases to give a thorough understanding of the mechanisms at work, and systematically provide pointers to the relevant literature when dealing with the general case would involve details that obscure the basic structures.

In our survey of linear logic, we will whenever possible take a purely syntactical point of view: this is not because we despise semantics, but because syntax, in particular for this logic, can be more easily accepted and understood by a wide public than its formal semantic underpinnings.

1.1 Linear Logic legitimate ambitions

Let us try to briefly explain why logic took part in the shaping of contemporary computer science, and how linear logic, by giving means to analyze the non-linear treatment of information, offers new tools and ideas in applying logic to computer science.

Logic and Computer Science

Modern logic began when people discussed the possibility of reading 'A implies B' as 'gimme A and I get you B'. They were beginning to suspect something. A proof of such a reading of the implicative statement might be more convincing than a proof of the traditional reading 'B is true whenever A is', and for good

reasons: it describes a process by which a proof of B can be produced from a proof of A. What better reason could we think of: it is because my proof of 'A implies B' will get you B, when you give me A, that it is a proof at all !

Logic, or at least proof-theory, sets up formal proof systems: intuitionistic predicate calculus, classical predicate calculus, arithmetics, higher order calculi ..., i.e., consistent and structured sets of process-building rules. Except that the 'processual' interpretation, if any, has to be made explicit.

Computer science, on the other hand, sets up computational mechanisms: application and substitution, polymorphism, exceptions, methods in object languages, message passing, eval/quote mechanism, variable assignment ... that is sets of process-building rules. Except that the logic of these processes, if any, has to made explicit.

At a given moment these two sciences met. People realized that the set of implication-only intuitionistic deductions was a core functional language called simply-typed lambda-calculus: the programmming language was a logic, the logic a programming language. This memorable meeting was called the 'Curry-Howard isomorphism'. Since then, 20 or so years ago, logic has ambitions in the field of programming: formal verification, formal semantics, language design, complexity theory ...

Linear Logic

Linear Logic begins with a further twist in the reading of 'A implies B': read now 'gimme *as many* A as I might need and I get you B'. The notion of *copy* which is so central to the idea of computation is now wired into the logic.

This opens up a new dimension, already intensely exploited, in applications of logic to computer science, and at the same time aims at a mathematical theory within which mathematics are comfortably done. It is both a theory and a tool.

New possibilities show up at two levels, which we might call the language level and the operational level. At the language level the way is open to have:

- *new formulas* expressing refined operational properties: 'gimme A once and I get you B'.

  Applications here range from knowledge representation in IA, refined Logic Programming where the born ability of Linear Logic to represent states is put to use, analysis of Classical Logic and computational interpretations thereof, namely exception mechanisms in programming languages, by means of embeddings in Linear Logic, refined temporal logics, linearity analysis.

- *new rules* expressing constraints on the use of copies resulting in a fragment of Linear Logic for polytime computations to mention only the most spectacular application.

- *new ways* of representing proofs displaying their hidden geometry.

  Here we think of Proof Nets, a parallel syntax for proofs which has been intensively studied, from the graph-theoretic viewpoint to the homological one.

  Proof-nets considerably ease the manipulation of syntax, and gave birth to explicit substitutions systems, themselves useful in the proof of compilers for functional languages, and were also used in linguistics to study parsing in natural languages, again to mention only a few applications.

At the operational level, the way proofs or processes are to be understood operationaly, that is how they are to be computed, there is room to have:

- *new evaluation schemes* of which the most salient aspect is their local and asynchronous treatment of copy.

  Optimal functional evaluation has been understood much better in the light of the 'geometry of interaction' a combinatorial model of proofs for closed computations. Compilations to parallel code are at hand.

- *new models of processes* capturing in a nice algebraic setting characteristics of evaluation: continuity, stability, and giving tools in proving correction and equivalence of programs.

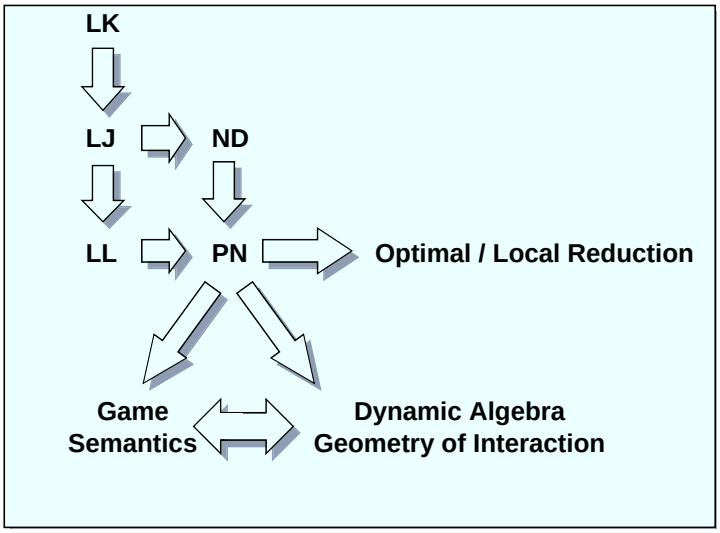
  Linear Logic has lead to the 'best' model of proofs as functions, namely hypercoherences, and has been of crucial help in the new 'games semantics', which solved the long-standing problem of 'full abstraction', that is strategies in those models all come from syntax. Linear Logic also yielded the first model of polymorphism, a logical system where code can be available for objects of different types, anticipating object-oriented programming. Progress has been made also in holding a topological model of programming, bridging the gap with classical mathematical methods.

1.2 Linear Logic and recent advances in theoretical computer science

Linear logic has shown great promise for theoretical computer science ever since its very birth, and over the past years part of these promises (as we will see,

mainly, but not exclusively, those concerning a fine analysis of sequential computation) have been fulfilled: optimal reduction, geometry of interaction, fully abstract semantics to cite but a few. In this book, we aim to give a broad overview of what has been accomplished, that should be both accessible for the newcomer and useful for the experienced researcher. Due to the novel nature of the subject, we believe that this goal is best accomplished by exposing in details the results and showing the techniques at work, even if in a simplified framework, rather than flooding the reader with the complex definitions and theorems needed to handle the more complex cases (for those readers that feel confident with the subject after this reading, we give full references to the research papers on the subject).

Here follows a schematic diagram showing the subjects treated in the book, and the chapters where they and the connections between them are treated: there is a clear path from classical sequent calculus (LK) to intuitionistic sequent calculus (LJ) to Linear logic (LL), and from intuitionistic natural deduction (ND) to Proof Nets (PN); then, in the lower part of the diagram we get to the more novel results concerning the algebrization of computation, optimal reduction, game semantics and explicit substitutions. Finally, we will give some taste of interesting fragments of linear logic using ELL.



1.3 Hilbert style or the axiomatic approach

blurb on the axiomatic approach to logical systems and its fitness to model theory.

Hilbert's axiomatic system for classical propositional logic

Here we recall Hilbert's system of axioms and rules for propositional logic. We have here only one *rule*, the "modus ponens"

$$\frac{A \Rightarrow B \quad A}{B}\, MP$$

and a set of axioms for each connective (recall that implication associates to the right, so $A \Rightarrow B \Rightarrow C$ really stands for $A \Rightarrow (B \Rightarrow C)$, and that implication has lower priority than all other connectives, so $A \wedge B \Rightarrow A \vee B$ really stands for $(A \wedge B) \Rightarrow (A \vee B)$):

Implication

$A \Rightarrow B \Rightarrow A$
$(A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow A \Rightarrow C$
$((A \Rightarrow B) \Rightarrow A) \Rightarrow A$ (Peirce's law)

Conjunction

$A \wedge B \Rightarrow A$
$A \wedge B \Rightarrow B$
$A \Rightarrow B \Rightarrow A \wedge B$

Disjunction

$A \Rightarrow A \vee B$
$B \Rightarrow A \vee B$
$(A \Rightarrow C) \Rightarrow (B \Rightarrow C) \Rightarrow A \vee B \Rightarrow C$

Falsity

$\bot \Rightarrow A$

Here a *proof* of a formula A is a sequence $A_0, \ldots A_n$ of formulae, where $A_n = A$ and each of the A_i is either an instance of an axiom or a consequence, via MP, of some formulae appearing earlier in the sequence (i.e. some A_j, A_k with $j, k < i$).

To see why this system is definitely not adequate for finding a proof, try to prove the trivial fact that $A \Rightarrow A$: you will face exactly the same difficulty as

that of programming the identity function using the well-known combinators S and K (see [Bar84, HS80] for more details on combinators and their relation to λ-calculus). For this reason the very first theorem one proves in this framework is the *deduction theorem*, that allows to get a proof of $A \Rightarrow B$ out of a proof of B under hypothesis A.

1.4 Gentzen style or the syntactic approach

blurb on Gentzen's works and the Hauptsatz, plus curry-howard isomorphisms.

A quite different approach is due to Gentzen, who introduced the notion of sequent calculus: a *sequent* is made up of two lists Γ and Δ of formulae separated by the *entailment* symbol $\vdash$, and is written $\Gamma \vdash \Delta$. The intuition is that Γ holds the hypothesis of an asserion and Δ the conclusions. In this presentation there only one axiom, the identity aiom $A \vdash A$, and to each connective are now associated no longer axioms, but *introduction rules*, on the left and on the right.

Let us see what the sequent calculus for classical logic looks like.

1.5 Classical Sequent Calculus

We will give a presentation of Classical Sequent Calculus where we insist upon its symmetries and show the importance of *structural* rules.

Rules of Sequent Calculus are divided into three groups.

Let $\Gamma, \Delta, \ldots$ stand for sequences of formulas.

Identity Rules

The *identity axiom* and the *cut rule*

$$\frac{}{A \vdash A}\, Id \qquad\qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\, Cut$$

simply say that A *is* A.

Structural rules

These rules are concerned with manipulation of contexts. By using them, we can access hypotheses (by *exchange*),

$$\frac{\Gamma, A, B, \Gamma' \vdash \Delta}{\Gamma, B, A, \Gamma' \vdash \Delta} LX \qquad \frac{\Gamma \vdash \Delta, A, B, \Delta'}{\Gamma \vdash \Delta, B, A, \Delta'} RX$$

discard hypotheses (by *weakening*)

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta} LW \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta} RW$$

or share hypotheses (by *contraction*)

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta} LC \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta} RC$$

Logical rules

The rules in this group introduce the logical connectors on the left or the right hand side, as in

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} L\wedge \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash A \wedge B, \Delta, \Delta'} R\wedge$$

The rule $L\wedge$ tells us that the $\wedge$ connective is just an internal notation for the comma on the left.
Similarly, we introduce the $\vee$ connective on the left or the right hand side:

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \vee B \vdash \Delta, \Delta'} L\vee \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta} R\vee$$

The rule $R\vee$ tells us that the $\vee$ connective is just an internal notation for the comma on the right.
Notice that we could as well define the introduction rules in another, more economic way:

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} L1\wedge \qquad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} L2\wedge \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta} R\wedge'$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta} L\vee' \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A \vee B, \Delta} R1\vee \qquad \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash A \vee B, \Delta} R2\vee$$

Notation 1.5.1 (Terminology) *These two distinct styles of managing contexts in the rules are called respectively* multiplicative *and* additive.

Of course, we have now rules for implication and negation, (and for quantifiers if we want to cover predicate calculus, but we will not deal with it here).

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta} L\neg \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta} R\neg$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \Rightarrow B \vdash \Delta, \Delta'} L\Rightarrow \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \Rightarrow B, \Delta} R\Rightarrow$$

Exercise 1.5.2 (additive cut) *Show that the cut rule Cut, in presence of structural rules, can be stated in a more economic, but equivalent way, as*

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta} Cut'$$

Exercise 1.5.3 *Show that in the classical (or intuitionistic*[1]*) framework $R\wedge$ and $R\wedge'$ are equivalent. Similarly, show that rule $L\wedge$ is equivalent to rules $L1\wedge$ and $L2\wedge$.*

Exercise 1.5.4 *Show that in the classical (or intuitionistic) framework $L\vee$ and $L\vee'$ are equivalent. Similarly, show that rule $R\vee$ is equivalent to rules $R1\vee$ and $R2\vee$.*

Remark 1.5.5 *Exercises 1.5.3 and 1.5.4 tell us that the choice we make in the presentation of classical (intuitionistic) sequent calculus are just a matter of taste. In Linear Logic, it will no longer be so!*

[1] But recall that in intuitionistic sequent calculus you have right to no more than one formula on the right!

Exercise 1.5.6 *Show that in the classical framework*[2] *the rules for* $\Rightarrow$ *can be derived from the rules for* $\neg$ *and* $\wedge$ *(or* $\neg$ *and* $\vee$*).*

Notice now that the rules for $\wedge$ are symmetrical with the rules for $\vee$: e.g. $R\wedge$ is the "mirror image" of $L\vee$. This is no surprise at all, since a classical sequent is symmetrical and these connectives are just notation for a comma: $\wedge$ on the left and $\vee$ on the right.
Another, deeper symmetry between the introduction and elimination rules for the same connective makes possible to replace a proof looking like

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma \vdash A, \Delta\end{array} \quad \begin{array}{c}\vdots\\ \Gamma' \vdash B, \Delta'\end{array}}{\Gamma, \Gamma' \vdash A \wedge B, \Delta, \Delta'}\, R\wedge \quad \dfrac{\begin{array}{c}\vdots\\ \Gamma'', A \vdash \Delta''\end{array}}{\Gamma'', A \wedge B \vdash \Delta''}\, L1\wedge}{\Gamma, \Gamma', \Gamma'' \vdash \Delta, \Delta', \Delta''}\, Cut$$

by the following one

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma \vdash A, \Delta\end{array} \quad \begin{array}{c}\vdots\\ \Gamma'', A \vdash \Delta''\end{array}}{\Gamma, \Gamma'' \vdash \Delta, \Delta''}\, Cut}{\Gamma, \Gamma', \Gamma'' \vdash \Delta, \Delta', \Delta''}\, \|$$

where the double bar means a sequence of structural rules (weakenings and exchanges).

Exercise 1.5.7 *Formulate such conversions for the three other cases:* $L\wedge/R\wedge$, $L\wedge/R\wedge$*' and* $L\,i\wedge/R\wedge$*'.*

Investigating carefully such conversions leads to the central property of viable logics, namely that it is possible to *eliminate the cut rule* from any proof which does not assume extra axioms (Gentzen's Hauptsatz). In other words if a sequent is provable at all then there exists a cut-free proof of it.

Exercise 1.5.8 (Subformula Property) *Show that any cut-free proof of a sequent* $\Gamma \vdash \Delta$ *without proper axioms only mentions subformulas of the formulae in* Γ *and* Δ*.*

[2] In the intuitionistic sequent calculus this fact is not true.

Remark 1.5.9 *The Intuitionistic Sequent Calculus can be obtained from the Classical Sequent calculus by constraining the number of formulae on the right of a sequent to at most one. This calculus also enjoys cut-elimination. In the intuitionistic case the process of cut-elimination is very similar to β-reduction in λ-calculus and promotes logic as a paradigm of computation (see [GLT90] for more details). In the classical case, the computational meaning of this process is still a matter of passionate investigation.*

Exercise 1.5.10 *Show that from any intuitionistic proof of $\vdash A \vee B$ one can extract a proof of A or a proof of B. Notice that this is the case because of the forbidding of the right contraction rule, implicit in the restriction on the right hand side of an intuitionistic sequent. So control over structural rules yields new properties; linear logic will impose the most severe control on these rules.*

Exercise 1.5.11 *State the natural conversion associated to contraction; one can see that this rule is responsible for the combinatorial explosion of cut-elimination.*

Notation 1.5.12 (Terminology) *It is common in the literature to find the following notation:*

LK, CSC the Classical Sequent Calculus

LJ, ISC the Intuitionistic Sequent Calculus

Weakening and contraction: bad bookkeepers

Now, all this is very nice, but before being satisfied with such a formalization of the laws of thought, let's look at the contraction and weakening rules from the point of view of daily life in a capitalistic world.
There are several reasons to get rid of weakening and contraction as structural rules, but we can start with a very simple and intuitive one. If we look at a sequent $\Gamma \vdash \Delta$ from the point of view of classical logic, it reads "from the truth of Γ follows the truth of Δ", but from a capitalistic point of view, we can read it as " if you give me the goods in Γ, I will give you one of the goods in Δ".
Now, classically, the contraction rule says that if you derive the truth of Δ using several occurrences of an assumption A, you can as well derive it with just one assumption: classical logics is interested in knowing whether A exists in the assumptions (i.e. wheter it is true or not), and not in how many times it occurs.
Similarly, weakening says that once you have the truth of Δ it does not harm to add some extraneous assumptions: Δ will stay true, and who cares some additional hypothesis.

Now, let's take the capitalistic point of view:
Contraction reads more or less this way: "if you can buy one of the goods in Δ using several goods A, then you can buy it with just one A".
And weakening becomes: "if you can buy one of the goods in Δ with Γ, then it does not harm to spend some more A's without getting anything more in change".
It is as easy to be convinced that the capitalistic reading would not be accepted by any sane merchant (that is not likely going to make you any present, as contraction would imply), nor by any careful buyer (that would not follow the advice given by weakening to waste carelessly his money for nothing).
Weakening and contraction as used in classical logic are bad bookkeepers: they make us loose control on our expenses, and hence waste our resources.
When the control of the use of some resource A is important, we need to do something to modify the capitalistic meaning of such structural rules: either we keep them (because we are fond of classical logic, and ready to waste our money for it) and we spend a lot of effort to formalize with a series of extra-logical axioms the fact that assuming A has a cost, or we can simply drop these rules, hoping to recover them in a simple way when dealing with resources that cost nothing.
Once we decide to be careful about resources, we start to be suspicious about the traditional $\wedge$ and $\vee$ connectives too, and we discover that each of them is now split in two. In fact, without weakening and contraction, we can no longer show equivalent the two formulations of the left introduction rules for $\wedge$ ($R\wedge$, $R\wedge'$) and $\vee$ ($L\vee$, $L\vee'$): actually exercises 1.5.3 and 1.5.4 require some weakenings and some contractions to be solved. This means that the different rules are describing different connectives, and we have now two conjunctions and two disjunctions.
This leads us to Linear Logic.

1.6 Linear Sequent Calculus

In Linear Logic, we have no longer the right to forget hypotheses or use them more than once, (no *weakening* and no *contraction*). This does not affect *Exchange* and *Identity* (while *cut* requires some care), but the classical $\wedge$ and $\vee$ connectives are now split in two.
We have *two* conjunctions: a *tensor product* (or *cumulative conjunction*: it corresponds to the "non-economic" presentation of the classical $\wedge$)

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

and a *direct product* (or *alternative conjunction*: it corresponds to the "economic" presentation of the classical $\wedge$):

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A\&B \vdash \Delta} L1\& \qquad \frac{\Gamma, B \vdash \Delta}{\Gamma, A\&B \vdash \Delta} L2\& \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A\&B, \Delta} R\&$$

Symmetrically, we get the rules for the *two* disjunctions: a *direct sum*: it corresponds to the "economic" presentation of the classical $\vee$):

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \oplus B \vdash \Delta} L\oplus \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A \oplus B, \Delta} R1\oplus \qquad \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash A \oplus B, \Delta} R2\oplus$$

and a *tensor sum*: it corresponds to the "non-economic" presentation of the classical $\vee$)

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

The rule for negation stays as in the classical case, and the negation operator (that is now noted $\perp$) is still involutive (i.e. $A^{\perp\perp}$ is the same as A).

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, A^{\perp} \vdash \Delta} L\perp \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash A^{\perp}, \Delta} R\perp$$

What about the cut rule then? Here we need some care, as the following remark and exercise show.

Remark 1.6.1 *The additive rule Cut' does not allow to internalize the $\vdash$ symbol: actually, it is not possible with such a cut rule to prove*

$$\frac{\vdash A \quad A \vdash B}{\vdash B}$$

Exercise 1.6.2 (Schellinx) *Show that adding the additive cut rule yields the same theory as adding the axiom $A \oplus A^{\perp}$; infer that this alternative cut cannot be eliminated.*

Hence, for Linear Logic we will keep the multiplicative cut rule Cut.
This presentation of a propositional fragment of Linear Logic based on symmetrical sequents is well suited to show how it can be derived from the classical frameworks by elimination of the *weakening* and *contraction* rules. It also allows us to define the *classical* and *intuitionistic* linear fragment in the same way as for the usual Gentzen calculus.

Notation 1.6.3 (Terminology) *In the literature, one can find the system above, with also rules for the constants and the* exponential *connectives, used to present what is usually called Classical Linear Logic (or CLL) and distinguish it from Intuitionistic Linear Logic (or ILL), where at most one formula is allowed to the right.*

Anyway, we can already see, just with these connectives, that this presentation of the system is very redundant: due to the symmetries (already pointed out in the classical case) present in the rules, there are many ways (using negation) to express some rules in terms of some other ones. This fact can be used to give a more concise presentation of the system, so let's restart from scrath and do the work more properly!

- We give atomic formulae in two forms: A and $A^\perp$. Then we say that the negation of A *is* $A^\perp$ and the negation $A^{\perp\perp}$ of $A^\perp$ *is* A.
- We *define* negation of non-atomic formulae by use of the De Morgan rules:

$$\begin{array}{rclcrcl} (A \otimes B)^\perp & = & A^\perp \parr B^\perp & \qquad & (A \& B)^\perp & = & A^\perp \oplus B^\perp \\ (A \parr B)^\perp & = & A^\perp \otimes B^\perp & & (A \oplus B)^\perp & = & A^\perp \& B^\perp \end{array}$$

- We *define* linear implication A $\multimap$ B as a shorthand for $A^\perp \parr$B.
- We convert a symmetric sequent $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ into an asymmetric right-only form by use of linear negation $\vdash A_1^\perp, \ldots, A_n^\perp, B_1, \ldots, B_m$

Exercise 1.6.4 *Check that the De Morgan's identities above correspond to linear equivalences in the two sided version of the calculus.*

Now, the identity axiom becomes $\vdash A^\perp, A$ and also subsumes at the same time the rules for linear negation. The cut rule is transformed into:

$$\frac{\vdash A, \Gamma \qquad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta}\, Cut$$

There is only one *exchange* rule that subsumes the left and right one of the symmetric presentation.

$$\frac{\vdash \Gamma, A, B, \Delta}{\vdash \Gamma, B, A, \Delta}\, X$$

and the logical rules are now expressed by:

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

$$\frac{\vdash A, \Gamma}{\vdash A \oplus B, \Gamma}1\oplus \qquad \frac{\vdash B, \Gamma}{\vdash A \oplus B, \Gamma}2\oplus \qquad \frac{\vdash A, \Gamma \quad \vdash B, \Gamma}{\vdash A \& B, \Gamma}\&$$

This is much better than the previous presentation! We can now continue with the full system of Propositional Linear Logic.
We introduce units:

$$1^\perp = \bot \qquad \bot^\perp = 1 \qquad \top^\perp = 0 \qquad 0^\perp = \top$$

$$\frac{\vdash \Gamma}{\vdash \bot, \Gamma}\bot \qquad \frac{}{\vdash 1}1 \qquad \text{(no rule for 0)} \qquad \frac{}{\vdash \top, \Gamma}\top$$

Exercise 1.6.5 *Check that actually* 1 *is an identity for* $\otimes$, $\bot$ *for* $\parr$, $\top$ *for* $\&$ *and* 0 *for* $\oplus$.

And finally, we allow two modalities $!A$ (*of course* A) and $?A$ (*why not* A) to deal with non-capitalistic resources:

$$(!A)^\perp = ?A^\perp \qquad (?A)^\perp = !A^\perp$$

$$\frac{\vdash \Gamma}{\vdash ?A, \Gamma}W? \qquad \frac{\vdash ?A, ?A, \Gamma}{\vdash ?A, \Gamma}C? \qquad \frac{\vdash A, \Gamma}{\vdash ?A, \Gamma}D? \qquad \frac{\vdash A, ?\Gamma}{\vdash !A, ?\Gamma}\,!$$

Exercise 1.6.6 (Terminology) *Prove the following linear equivalences:*

$$!(A \& B) \equiv (!A) \otimes (!B) \quad ?(A \oplus B) \equiv (?A) \parr (?B).$$

So that is the reason why $\otimes$ *and* $\parr$ *are called* multiplicatives*, while the* $\&$ *and* $\oplus$ *are called* additives *and* $!$ *and* $?$ exponentials*.*

Exercise 1.6.7 *Show that the aforementioned fundamental isomorphisms are indeed isomorphisms of vector spaces if read with the following dictionary:* $E^\perp$ *is the dual of* E, $\otimes$ *and* $\parr$ *are both the tensor product,* $\&$ *and* $\oplus$ *the direct sum and both modalities the symmetric algebra.*

Remark 1.6.8 *The rule for ! imposes constraints on the context, as the rule for &. The rules for exponentials carry some resemblance to the rules for the modal connectives for* necessity *and* possibility *($\Box$ and $\Diamond$).*

Notation 1.6.9 (Terminology) *It is common in the literature to find the following names for different fragments of Linear Logic.*

LL The full propositional fragment of Linear Logic

MLL The multiplicative fragment of Linear Logic (no exponentials)

MALL The multiplicative and additive fragment of Linear Logic (no exponentials)

MELL The multiplicative fragment of Linear Logic with exponentials.

Linear logic also enjoys cut-elimination and subformula property: state the conversions (reduction moves) for each pair of dual connectives : $\otimes$ and $\parr$, $\&$ and $\oplus$, ? and !.
Cut-elimination is now (as opposed to the case of classical logic) a deterministic process; and indeed full LL can be seen as a computationaly sound completion of MALL.

Exercise 1.6.10 *Notice that the commas are multiplicative: show that $\Gamma \vdash \Delta$ if and only if $\otimes\Gamma \vdash \parr\Delta$. Does the same hold for $\&\Gamma \vdash \oplus\Delta$?*

Exercise 1.6.11 (invertible connectives) *Prove that the $\parr$ is invertible, i.e., that $\Gamma, A \parr B$ is provable iff Γ, A, B is provable.*
State and prove (resp. disprove) a similar property for the $\&$ (resp. $\otimes$ and PLUS).
Prove also that the ! is invertible if all formulae of the context are prefixed by a ? modality.

Exercise 1.6.12 (Unprovable formulae)

1. *Show that the following sequents are unprovable in LL, although they are classically valid substituting $\wedge$ to $\otimes$ (check).*

$$\vdash A \otimes A, A^{\perp} \otimes A^{\perp} \text{ and } \vdash A \otimes B, A^{\perp} \otimes B, A \otimes B^{\perp}, A^{\perp} \otimes B^{\perp}$$

$$\vdash (A \otimes A) \multimap A \text{ and } \vdash A \multimap (A \otimes A)$$

2. *Show also that* $\vdash A^{\perp} \oplus A$ *is not provable. So it is of no interest to define an additive implication: compare with 1.6.1.*

Exercise 1.6.13 (Girard) *Suppose one adds a new pair of modalities, !', ?', with the same rules as for the standard ones. Show you cannot prove the equivalence of this new pair with the old one.*

Now let's see that the rules for exponentials capture exactly *weakening* and *contraction*.

Exercise 1.6.14 (Schellinx)

1. *Show that if* $A \multimap (1 \& (A \otimes A))$ *then left* weakening *and* contraction *rules are* admissible *for A.*
2. *Show that if* $A \multimap !\, A$ *then* $A \multimap (1 \& (A \otimes A))$.
3. *Open problem: does the converse hold? Try to state the problem.*

Exercise 1.6.15 (Joinet) $*$

1. *Prove that* $!!A$ *is linearly equivalent to* $!A$.
2. *Prove there are only seven modalities (combinations of the two modalities ! and ?) up to linear equivalence, and draw the lattice of their relationships.*

Exercise 1.6.16 (Danos-Joinet-Schellinx) $*$
The purpose of this long exercise is to state an admissible extension of the contraction-rule in LL.
We shall use $\diamond A$ *to denote one of* A *and* $?A$*, and* $\bullet$ *to denote any of the connectives* $\otimes, \parr, \&, \oplus$.
Let the notions of positive, negative subformulas of a linear formula F *be given as usual. Say that a linear formula* G *is a positive (resp. negative) decoration of* F *iff* G *has been obtained from* F *by questioning some (maybe none) positive (resp. negative) subformulas of* F .
a) Let F *be any formula,* F^{+} *be any positive and* F^{-} *any negative decoration of* F*. By induction on* F *show:* $F \vdash F^{+}$ *and* $F^{-} \vdash F$.
b) Show that one turns the set $D+(F)$ *of all positive decorations of* F *into a lattice with the ordering* $A \leq B$ *iff the decoration of* B *is equal to or extends that of* A*; the join* $A \sqcup B$ *of two elements* A *and* B *being the surperposition of both decorations.*

c) So by a) $A \leq B$ entails $B \vdash A$, is the converse true ?
d) show with the help of a) that the "extended" dereliction:

$$\frac{\Gamma \vdash F', F'', \Delta}{\Gamma \vdash \diamond(F' \sqcup F'') \vdash \Delta}$$

where F', F'' are in $D^+(F)$ and $\diamond = ?$ iff neither F' nor F'' is questioned, is admissible.

Exercise 1.6.17 (Girard) *Assign an integer $v(p_i)$ in $\mathbf{Z}$ to every atomic symbol p_i, and then extend this valuation to non atomic formulae as follows:*

$$v(p_i^\perp) = 1 - v(p_i) \qquad v(1) = 1 \qquad v(\bot) = 0$$

$$v(A \otimes B) = v(A) + v(B) - 1 \qquad v(A \parr B) = v(A) + v(B).$$

1. *find a relation between $v(A)$ and $v(A^\perp)$*
2. *show that if A is provable then v(A) = 1*
3. *find a formula A (without 1 or $\bot$) that is not provable and such that $v(A) = 1$ for every possible valuation v.*

Exercise 1.6.18 (Girard) *Let's focus now on those formulae of MALL that are closed, i.e. the ones built out of 1 and $\bot$ only. Consider now the classical valuation $c(1) = \top, v(\bot) = \bot$ extended to non atomic formulae as follows:*

$$v(A \otimes B) = c(A) \wedge c(B) \qquad c(A \parr B) = c(A) \vee v(B).$$

1. *show that $1 \parr 1$, $\bot$ are not provable, and deduce that $\bot \otimes (1 \parr 1)$ is not provable*
2. *show a closed formula A that is not provable, classically false (i.e. $c(A) = \bot$) and such that $v(A) = 1$; show also a classically true formula B such that $v(B) \neq 1$.*
3. *find for every $n \in \mathbf{Z}$ a closed formula A_n such that $v(A_n) = n$ and $c(A_n) = \top$*
4. *show a non provable closed formula A s.t. $v(A) = 1$ and $c(A) = \top$*

Exercise 1.6.19 (Girard) *We still stay in the world of closed formulae.*

1. *show a formula A s.t. $(A \otimes A^\perp) \parr 1$ is not provable*

2. *given any set E and a valuation $\phi(A) \in E$ for all closed formulae A that enjoys the following properties:*

   - *if $\phi(A) = \phi(B)$ then $\phi(A^\perp) = \phi(B^\perp)$*
   - *if $\phi(A) = \phi(A')$ and $\phi(B) = \phi(B')$ then $\phi(A \otimes B) = \phi(A' \otimes B')$*

   Show that for every $e \in E$ the set of closed formulae A s.t. $\phi(A) = e$ is different from the set of provable formulae.

3. *in particular, recover the results of 1.6.18, i.e. the fact that $v(A) = 1$ and $c(A) = \top$ are not sufficient conditions for provability*

Exercise 1.6.20 (Danos, Exploring the additive-only world.) *1. Prove that in the purely additive fragment sequents have exactly one formula on the lefthand side and one on the righthand side.*

2.

   a) Prove that the cut-free additive calculus is decidable (hint: all rules have their premisse(s) smaller than the conclusion).

   b) Deduce from the general cut-elimination result the cut-elimination for the additive fragment. One just has to check that conversions stay within the fragment.

3. *Facts about provability:*

   a) Prove both additives are associative and commutative.

   b) Prove $A\&B \vdash A \oplus B$ and disprove the converse (hint: use cut-elimination).

   c) Prove $A \oplus B \vdash C$ holds iff $A \vdash C$ and $B \vdash C$, and the symmetric statement for $C \vdash A\&B$. One says $\oplus$ is reversible on the left and $\&$ on the right.

   d) Prove $A\&B \vdash C \oplus D$ holds iff one of the four following sequent holds: $A \vdash C \oplus D$, $B \vdash C \oplus D$, $A\&B \vdash C$, $A\&B \vdash D$.

4. *We now want to interpret our additives as traditional mathematical operations; since they look like union and intersection, we might have a look at lattices.*

   Let a set X of propositional variables be given. Let L be a lattice generated by X, that is an ordering with infima and suprema with a mapping $(.)^\circ$ from X to L.

Any formula A can be interpreted as an element A° of L as follows: $(A \oplus B)^\circ = A^\circ \vee B^\circ$, $(A\&B)^\circ = A^\circ \wedge B^\circ$.

Any sequent $A \vdash B$ can then be interpreted as $A^\circ \leq B^\circ$, an atomic statement about the ordering in L. This statement might be right and might be wrong.

a) Show that if $A \vdash B$ is provable then $A^\circ \leq B^\circ$ holds in L. Did this requires L to be a lattice? What if one restricts to the plus-only fragment of ALL.

b) Show the converse when L is freely generated by X, which means for all L' generated by X with mapping $(.)^{\circ\prime}$ there is a unique lattice morphism f such that $(.)^\circ; f = (.)^{\circ\prime}$. It is enough to show that (ALL-formulas/$\Leftarrow\Rightarrow$,$\vdash$) is a lattice.

c) Thus distributivity must fail in general. Prove it directly, then prove it by building a non-distributive lattice (hint: take generators incomparable, a top and a bottom). Formula models are often used to refute sequents.

d) Suppose one adds some non-logical axioms of the form $p \vdash q$, for some p's and q's in X, and requires that $p^\circ \leq q^\circ$ when $p \vdash q$. Reconsider the last two points in this case.

1.7 Complexity Issues

The last point of the exercise above tells us that there is no hope to find a "simple" valuation catching multiplicative truth. This is a hint that multiplicative provability is a difficult problem (in terms of complexity). An extensive research work has been dedicated to the study of the complexity and decidability issues for several fragments of LL. While a recent presentation of the current knowledge can be found in [Lin95], we summarize here a few relevant results:

MLL

- provability is NP-complete [Kan94b]

MALL

- provability is PSPACE-complete [LMSS92]
- LL provability is undecidable [LMSS92]

Surprisingly (or not? didn't we say that the expressive power of LL comes from the built-in control of resources, not from the combinatorics of some truth valuation function?) these results stay the same even if we focus on the fragments

of the logics where only the constants, and no propositional variables are allowed [Kan94a, LW94]: indeed, it is possible to encode arbitrary formulae into constant-only formulae preserving provability.

1.8 Informal Semantics

Before starting our study of Linear Logic, that involves a series of relevant syntactic results, we will now try to give an informal semantics of Linear Logic, and try to understand what these linear connectives are by using that capitalistic point of view that already helped us to be suspicious about the *weakening* and *contraction* rules, otherwise so innocuous-looking.
This will help us to familiarize with Linear Logic, and will hopefully convey enough intuitive meaning to make these new entities $\otimes$, $\parr$, $\&$, $\oplus$ etc. as natural as (or more natural than) the usual classical connectives.

Linear connectives: useful devices to read your menu

Linear implication and the four basic connectives that linear logic supllies in change of the traditional $\wedge$ and $\vee$ are really nothing new to the people that is used to go frequently to restaurants that offer fixed-price menu's. Let's have a look at a typical one (as in [Gir90]).
We can easily formulate this menu in linear logic as

$$75\mathrm{FF}^{\perp} \parr (Q \& S) \otimes (P \& F) \otimes ((B \oplus R \oplus O \oplus A) \& (M \& G \& T))$$

The $\parr$ connective tells us that if we give in 75 Frs (that sadly disappear immediately afterwards: the fact that we give in the francs is expressed by the little $\perp$) for one "entree" and one "plat" and either a "dessert" or some "fruit". So a $\parr$ connective somewhere in a formula tells us that we are faced with a trading situation: $A^{\perp} \parr$B means we can get A or B in the very particular sense that if we got an A somewhere (we are not too poor) either we keep this A or we get a B by exchanging it with our A. We cannot get both, because we live in a capitalistic world. On the other hand, we are pretty sure that we will get both one "entree" and a "plat", that is formalised by the $\otimes$ linear connective.
Now, let's keep looking at the menu. As an "entree" we can have a "quiche lorraine" or a "saumon fume", but not both. Here, anyway, the connective we use is not a $\parr$, but a $\&$: we are in a different situation that allows us to get a "quiche lorraine" or a "saumon fume", at our will, but to get the "quiche lorraine" we are not obliged to give in a "saumon fume". After all, we went in the restaurant to get some food, not to give food away! Similarly for the "plat". Then we can choose

Menu a 75 Frs
Entree
quiche lorraine
ou
saumon fume
et
Plat
pot-au-feu
ou
filet de canard
et
Fruit
selon saison (banane ou raisin ou oranges ou ananas)
ou
Dessert au choix (mistere, glace, tarte aux pommes)

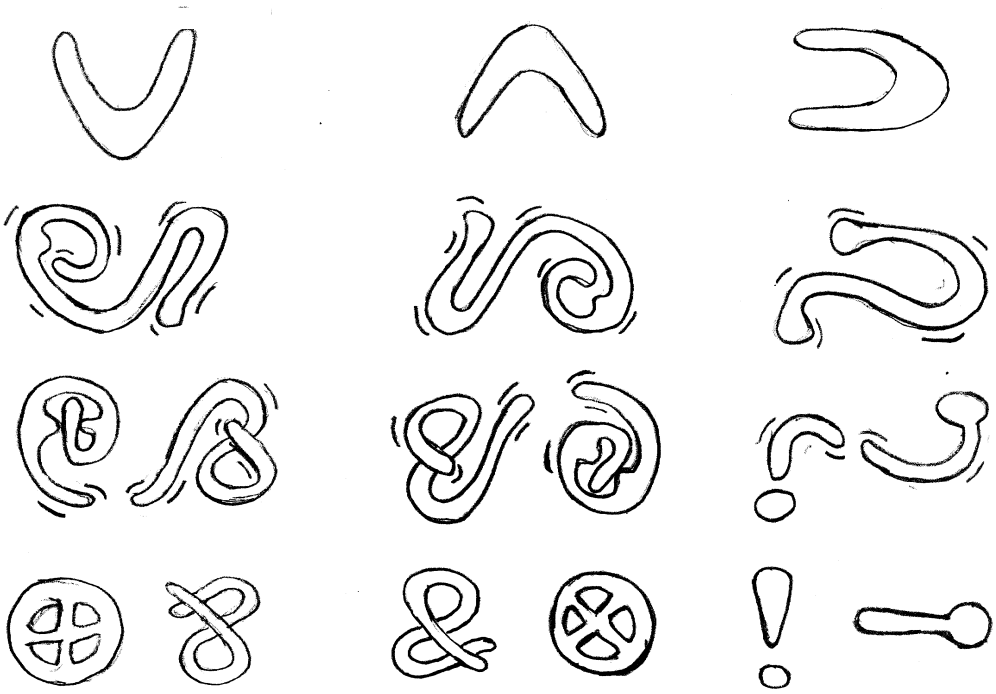
Figure 1.1: The linear Menu

to have one (but only one) between a "Dessert" or a "fruit" (again a $\&$), but here some other peculiar phenomenon happens: we know that we can get only one of the fruits listed in the Menu, but now we cannot decide which one, not even by trading something in exchange. Hence we cannot use a $\otimes$ to formalize it (we get only one fruit, not all), nor a $\&$ or a $\parr$ (we cannot choose the one we want, not even by trading), so here comes to our rescue the last connective of linear logic: the $\oplus$. This connective tells us that the object we will get from $A \oplus B$ is either an A or a B, but nothing more.

CHAPTER 2

Logical Embeddings into Linear Logic

citazione sistemi logici
Jean-Yves Girard
????



2.1 Recovering Intuitionistic Logic

The exponential connectives allow us to recover the full power of Intuitionistic Logic: we can provide a faithful translation * of intuitionistic formulae and proofs into linear ones, in such a way that if a sequent $\Gamma \vdash A$ of LJ is provable, then its translation $\Gamma^\star \vdash A^\star$ is provable.
Notice that, for simplicity, we will give the translation just for LJ *without* the right weakening rule: this fragment is largely sufficient both to encode the λ-calculus without explicit escape constructs and to faithfully encode the full classical sequent calculus, as we will see in the next section. If one really wants to handle also the right weakening, then an LJ-sequent $\Gamma \vdash A$ should be translated into something like $!\Gamma \vdash ?!A$: we do not consider this extra burden worthwhile, and suggest that the interested reader use the translation of LK into LL instead[1].

Translation of formulae.

If A is atomic, then $A^\star = A$.

$$\begin{array}{lcl}
(A \Rightarrow B)^\star & = & ?(A^\star)^\perp \parr !B^\star \;=\; (!A^\star) \multimap !B^\star \\
(A \wedge B)^\star & = & !A^\star \& !B^\star \\
(A \vee B)^\star & = & !A^\star \oplus !B^\star \\
(\neg A)^\star & = & ?(A^\star)^\perp
\end{array}$$

Remark 2.1.1 *Exponentials are used in the translation; they are in charge of non linear effects. The first equation is remarkable, it is as if linear logic were a looking-glass through which the intuitionistic implication appears as a compound connective. Notice that the translation suggested for the conjunction is for its additive presentation: if you want to take the multiplicative presentation, than you should better try* $!A \otimes !B$.

Conventions for sequents in LL.

Let's write $\Gamma \vdash \Delta$ for $\vdash \Gamma^\perp, \Delta$.
The idea behind this translation from LJ to LL is as follows:

$$\begin{array}{ccc}
\Pi & & \Pi^\star \\
\vdots & & \vdots \\
\Gamma \vdash A & \overset{\star}{\rightsquigarrow} & !\Gamma^\star \vdash !A^\star
\end{array}$$

[1] It is also possible to give a translation where ? is used only for weakening, see [Jac94].

Now we define a translation of LJ-proofs by induction on the structure of proofs in LJ.

axiom

$$\frac{}{A \vdash A} \quad \overset{\star}{\rightsquigarrow} \quad \frac{}{!A^\star \vdash !A^\star}$$

cut

$$\frac{\begin{array}{cc}\Pi_1 & \Pi_2 \\ \vdots & \vdots \\ \Gamma, A \vdash C & \Gamma' \vdash A\end{array}}{\Gamma, \Gamma' \vdash C} \quad \overset{\star}{\rightsquigarrow} \quad \frac{\begin{array}{cc}\Pi_1^\star & \Pi_2^\star \\ \vdots & \vdots \\ !\Gamma^\star, !A^\star \vdash !C^\star & !\Gamma'^\star \vdash !A^\star\end{array}}{!\Gamma, !\Gamma'^\star \vdash !C^\star}$$

weakening

$$\frac{\begin{array}{c}\Pi \\ \vdots \\ \Gamma \vdash C\end{array}}{\Gamma, A \vdash C} \quad \overset{\star}{\rightsquigarrow} \quad \frac{\begin{array}{c}\Pi^\star \\ \vdots \\ !\Gamma^\star \vdash !C^\star\end{array}}{!\Gamma^\star, !A^\star \vdash !C^\star}$$

contraction

$$\frac{\begin{array}{c}\Pi \\ \vdots \\ \Gamma, A, A \vdash C\end{array}}{\Gamma, A \vdash C} \quad \overset{\star}{\rightsquigarrow} \quad \frac{\begin{array}{c}\Pi^\star \\ \vdots \\ !\Gamma^\star, !A^\star, !A^\star \vdash !C^\star\end{array}}{!\Gamma^\star, !A^\star \vdash !C^\star}$$

left arrow

$$\frac{\begin{array}{cc}\Pi_1 & \Pi_2 \\ \vdots & \vdots \\ \Gamma, B \vdash C & \Gamma' \vdash A\end{array}}{\Gamma, \Gamma', A \Rightarrow B \vdash C} \quad \overset{\star}{\rightsquigarrow} \quad \frac{\dfrac{\begin{array}{cc}\Pi_1^\star & \Pi_2^\star \\ \vdots & \vdots \\ !\Gamma^\star, !B^\star \vdash !C^\star & !\Gamma^\star \vdash !A^\star\end{array}}{!\Gamma^\star, !\Gamma'^\star, (!A^\star) \multimap (!B^\star) \vdash !C^\star}}{!\Gamma^\star, !\Gamma'^\star, !((!A^\star) \multimap (!B^\star)) \vdash !C^\star}$$

right arrow

$$\frac{\begin{array}{c}\Pi \\ \vdots \\ \Gamma, A \vdash C\end{array}}{\Gamma \vdash A \Rightarrow C} \quad \overset{\star}{\rightsquigarrow} \quad \frac{\dfrac{\begin{array}{c}\Pi^\star \\ \vdots \\ !\Gamma^\star, !A^\star \vdash !C^\star\end{array}}{!\Gamma^\star \vdash (!A^\star) \multimap (!C^\star)}}{!\Gamma^\star \vdash !((!A^\star) \multimap (!C^\star))}$$

Exercise 2.1.2 *Complete the translation of proofs treating the case of* $\wedge$, $\vee$ *and* $\neg$.

Remark 2.1.3 (Decoration) *Indeed, the translation of proofs we have exhibited has a very nice structural property (if you properly treat the cases of disjunction and conjunction in the previous exercise): if we forget about the exponentials and the eventually duplicated sequents, the underlying tree of the derivation (often called a* skeleton*) is the same in the* LJ*-proof and in the translated* LL*-proof. Indeed, one can see the whole translation as a process of* decoration *of the intuitionistic proof with a bunch of exponentials. This is not the case, for example, of the more economical and well-known translation that we will present later.*

The translation of proofs we have just exhibited actually proves the following

Theorem 2.1.4 (Soundness) *Let* A *be an intuitionistic propositional formula of* LJ*. If* A *is provable in* LJ*, then* $A^\star$ *is provable in* LL*.*

2.2 More economical embeddings

The translation we just presented is often referred to as the "plethoric" translation , as it uses much more exponentials than what is strictly necessary (though it is indeed minimal if one asks for a uniform decoration procedure as defined above). Here we present another, more economical embedding, which is better known in the linear logic literature, but it should be noted that it is possible to give an embedding which is the most economical among all embeddings, as shown in [DJS93, DJS95a], and could be useful in compiling into a linear machine, as suggested in [Rov92]. Unlike the plethoric translation, though, this new translation will not respect the skeleton of a proof, most notably in the case of the arrow, which we will detail here.

Translation of formulae.

If A is atomic, then $A^* = A$.

$$\begin{array}{lcl}
(A \Rightarrow B)^* & = & ?(A^*)^\perp \parr B^* \;=\; (!A^*) \multimap B^* \\
(A \wedge B)^* & = & A^* \& B^* \\
(A \vee B)^* & = & !A^* \oplus !B^* \\
(\neg A)^* & = & ?(A^*)^\perp
\end{array}$$

The idea behind this translation from LJ to LL is as follows:

$$\begin{array}{c}\Pi\\ \vdots\\ \Gamma \vdash A\end{array} \quad \overset{*}{\rightsquigarrow} \quad \begin{array}{c}\Pi^*\\ \vdots\\ !\Gamma^* \vdash A^*\end{array}$$

Exercise 2.2.1 *Find a proof* $\Phi(X,Y)$ *in* LL *of* $!((!X) \multimap Y) \vdash (!X) \multimap (!Y)$ *and of* $!(X \& Y) \vdash (!X) \& (!Y)$.

Remark 2.2.2 (Savings balance) *It is interesting to remark that only on the connectives* $\multimap$ *and* $\wedge$ *we could save on exponentials, while you can show that such a savings is not really possible on the (notoriously proof-theoretically ugly)* $\vee$ *and* $\neg$ *connectives (that would imply, for example, to be able to prove something like* $!(A \oplus B) \vdash !(!A \oplus !B)$*, which is not likely). Indeed, one can really show that such savings are only possible on the connectives that are right-invertibles (see [DJS95b] for more details).*

Now we define the translation by induction on the structure of proofs in LJ.

axiom

$$\frac{}{A \vdash A} \quad \overset{*}{\rightsquigarrow} \quad \frac{\dfrac{}{A^* \vdash A^*}}{!A^* \vdash A^*}$$

cut

$$\frac{\begin{array}{cc}\Pi_1 & \Pi_2\\ \vdots & \vdots\\ \Gamma, A \vdash C & \Gamma' \vdash A\end{array}}{\Gamma, \Gamma' \vdash C} \quad \overset{*}{\rightsquigarrow} \quad \frac{\begin{array}{cc}\Pi_1^* & \Pi_2^*\\ \vdots & \vdots\\ !\Gamma^*, !A^* \vdash C^* & \dfrac{!\Gamma'^* \vdash A^*}{!\Gamma'^* \vdash !A^*}\end{array}}{!\Gamma, !\Gamma'^* \vdash C^*}$$

left arrow

$$\dfrac{\dfrac{\Pi_1}{\vdots}\ \Gamma, B \vdash C \qquad \dfrac{\Pi_2}{\vdots}\ \Gamma' \vdash A}{\Gamma, \Gamma', A \Rightarrow B \vdash C} \quad \overset{*}{\rightsquigarrow} \quad \dfrac{\Phi(A^*, B^*) \quad \dfrac{\dfrac{\Pi_1^*}{\vdots}\ !\Gamma^*, !B^* \vdash C^* \qquad \dfrac{\dfrac{\Pi_2^*}{\vdots}\ !\Gamma^* \vdash A^*}{!\Gamma^* \vdash\ !A^*}}{!\Gamma^*, !\Gamma'^*, (!A^*) \multimap (!B^*) \vdash C^*}}{!\Gamma^*, !\Gamma'^*, !((!A^*) \multimap B^*) \vdash C^*}$$

right arrow

$$\dfrac{\dfrac{\Pi}{\vdots}\ \Gamma, A \vdash C}{\Gamma \vdash A \Rightarrow C} \quad \overset{*}{\rightsquigarrow} \quad \dfrac{\dfrac{\Pi^*}{\vdots}\ !\Gamma^*, !A^* \vdash C^*}{!\Gamma^* \vdash\ !A^* \multimap C^*}$$

Exercise 2.2.3 *Complete the translation treating the case of* $\wedge$ *and* $\vee$.

The translation of proofs we have just exhibited actually proves the following

Theorem 2.2.4 (Soundness) *Let* A *be an intuitionistic propositional formula of* LJ. *If* A *is provable in* LJ, *then* A^* *is provable in* LL.

Exercise 2.2.5 *Try the translation:* $(A \wedge B)^* = A^* \otimes B^*$.

Completeness of the translation

We discuss here a simple proof of completeness of the translation for the propositional fragment of LJ with the connectives $\Rightarrow$ $\vee$ and $\wedge$, but no constants (like true and false) and no negation.

1. Recall that ILL, the Intuitionistic Linear Logic, is obtained from the symmetric presentation of LL by restricting the formulae on the right hand side of a sequent to at most one. Furthermore, we need only consider here the connectives: $\multimap$, $\&$, $\oplus$, and $!$. (Why?).

2. Show that if Π is a cut-free proof in LL of $\Gamma \vdash A$ where Γ and A are without $^\perp$, $\parr$ or constants, then it stays wholly in ILL. (hint: inspect each rule that can be performed on such a sequent and show they all are in ILL; in particular, this means checking that any rule that can be applied to derive in LL a sequent with exactly one consequent $\Gamma \vdash A$ can use as premisses

only sequents with one consequent, hence still in ILL. In particular, notice that if a premiss of a left $\multimap$ rule has an empty succedent, then this premiss cannot be derived in the system without linear negation and constants).

3. Deduce from these facts the

Theorem 2.2.6 (Completeness) *Let A be an intuitionistic propositional formula of LJ, If $A^\star$ is provable in LL, then it is provable in LJ.*

Proof. Hint: show that the following read-back procedure $^\circ$ from ILL to LJ is *correct*.

If A is atomic, then $A^\circ = A$.

$$\begin{array}{lcl} (!A)^\circ & = & A^\circ \\ (A \multimap B)^\circ & = & A^\circ \Rightarrow B^\circ \\ (A \& B)^\circ & = & A^\circ \wedge B^\circ \\ (A \oplus B)^\circ & = & A^\circ \vee B^\circ \end{array}$$

□

Remark 2.2.7 (Schellinx) *Notice, though, that if we extend the translation to the full LJ system with the $\bot$ constant, and hence negation defined in the usual way, it is then rather more complex to show that this translation is complete. Schellinx observed that in the presence of the rules for 0, a derivation of an ILL sequent using only the connectives of ILL could have more than one consequent, as in the following case:*

$$\dfrac{\dfrac{\dfrac{A \vdash A \quad 0 \vdash A, A \multimap 0}{A, A \multimap 0 \vdash A, A \multimap 0} \quad 0 \vdash}{A, A \multimap 0, A \multimap 0 \vdash A \multimap 0}}{!A, !(A \multimap 0), !(A \multimap 0) \vdash A \multimap 0}$$
$$!A, !(A \multimap 0) \vdash A \multimap 0$$

Hence the proof we conducted above is no longer correct. The theorem is nonetheless true, and the interested reader is referred to [Sch91] for details on a correct, but more complex proof. It is possible to generalize the theorem to the full calculus with (first or second order) quantifiers.

2.3 Recovering Classical Logic

In this section, we will show how classical logic can be full and faithfully embedded into LL. To do so, we choose not to directly give the usual translation, but rather to obtain it via an embedding of LK into LJ which is also well behaved w.r.t. to the skeleton of a classical proof. As the reader will immediately notice, this does not mean that the skeleton is exactly the same, but only up to some intervening negations that are instrumental to keep the translation inside LJ.

The traditional embeddings: Gödel's translation

Definition et discussion du fait que cela ne marche pas trop bien parce que pas substitutif et pas preserve cut-free et modifie l'arbre (pas decoratif).

A decorative embedding of LK into LJ

We define a translation $\circ$ from classical formulae into intuitionistic formulae as follows:
If A is atomic, then $A^\circ = A$.

$$\begin{array}{lcl} (A \to B)^\circ & = & (\neg B^\circ) \Rightarrow (\neg A^\circ) \\ (A \wedge B)^\circ & = & (\neg\neg A^\circ \wedge \neg\neg B^\circ) \\ (A \vee B)^\circ & = & (\neg\neg A^\circ \vee \neg\neg B^\circ) \\ (\neg A)^\circ & = & \neg A^\circ \end{array}$$

The idea behind this translation from LK to LJ is as follows:

$$\begin{array}{ccc} \Pi & & \Pi^\circ \\ \vdots & & \vdots \\ \Gamma \vdash \Delta & \overset{\circ}{\rightsquigarrow} & \Gamma^\circ, \neg\Delta^\circ \vdash \end{array}$$

Remark 2.3.1 *This actually embeds LK in a fragment of LJ where no right weakening is used, and where all right introduction rule is immediately followed by a negation rule that moves the introduced formula to the left. We will note this fragment* $LJ^{\neg}$.

Now we define a translation of derivations by induction on the structure of proofs in LK.

axiom

$$\frac{}{A \vdash A} \quad \overset{\circ}{\rightsquigarrow} \quad \frac{\dfrac{}{A^\circ \vdash A^\circ}}{A^\circ, \neg A^\circ \vdash}$$

cut

$$\frac{\begin{array}{cc}\Pi_1 & \Pi_2 \\ \vdots & \vdots \\ \Gamma, A \vdash \Delta & \Gamma' \vdash A, \Delta'\end{array}}{\Gamma, \Gamma' \vdash \Delta, \Delta'} \quad \overset{\circ}{\rightsquigarrow} \quad \frac{\dfrac{\begin{array}{c}\Pi_1^\circ \\ \vdots \\ \Gamma^\circ, A^\circ, \neg\Delta^\circ \vdash\end{array}}{\Gamma^\circ, \neg\Delta^\circ \vdash \neg A^\circ} \quad \begin{array}{c}\Pi_2^\circ \\ \vdots \\ \Gamma'^\circ, \neg A^\circ, \neg\Delta'^\circ \vdash\end{array}}{\Gamma^\circ, \Gamma'^\circ, \neg\Delta^\circ, \neg\Delta'^\circ \vdash}$$

right weakening

$$\frac{\begin{array}{c}\Pi \\ \vdots \\ \Gamma \vdash \Delta\end{array}}{\Gamma \vdash A, \Delta} \quad \overset{\circ}{\rightsquigarrow} \quad \frac{\begin{array}{c}\Pi^\circ \\ \vdots \\ \Gamma^\circ, \neg\Delta^\circ \vdash\end{array}}{\Gamma^\circ, \neg A^\circ, \neg\Delta^\circ \vdash}$$

right contraction

$$\frac{\begin{array}{c}\Pi \\ \vdots \\ \Gamma \vdash A, A\Delta\end{array}}{\Gamma \vdash A, \Delta} \quad \overset{\circ}{\rightsquigarrow} \quad \frac{\begin{array}{c}\Pi^\circ \\ \vdots \\ \Gamma^\circ, \neg A^\circ, \neg A^\circ, \neg\Delta^\circ \vdash\end{array}}{\Gamma^\circ, \neg A^\circ, \neg\Delta^\circ \vdash}$$

left arrow

$$\dfrac{\begin{array}{cc}\Pi_1 & \Pi_2\\ \vdots & \vdots\\ \Gamma, B \vdash \Delta & \Gamma' \vdash A, \Delta'\end{array}}{\Gamma, \Gamma', A \to B \vdash \Delta, \Delta'} \quad \overset{\circ}{\rightsquigarrow} \quad \dfrac{\dfrac{\begin{array}{c}\Pi_1^\circ\\ \vdots\\ \Gamma^\circ, B^\circ, \neg\Delta^\circ \vdash\end{array}}{\Gamma^\circ, \neg\Delta^\circ \vdash \neg B^\circ} \quad \begin{array}{c}\Pi_2^\circ\\ \vdots\\ \Gamma^\circ, \neg A^\circ, \neg\Delta'^\circ \vdash\end{array}}{\Gamma^\circ, \Gamma'^\circ, (\neg B^\circ) \Rightarrow (\neg A^\circ), \neg\Delta^\circ, \neg\Delta'^\circ \vdash}$$

right arrow

$$\dfrac{\begin{array}{c}\Pi\\ \vdots\\ \Gamma, A \vdash B, \Delta\end{array}}{\Gamma \vdash A \to B, \Delta} \quad \overset{\circ}{\rightsquigarrow} \quad \dfrac{\dfrac{\dfrac{\begin{array}{c}\Pi^\circ\\ \vdots\\ \Gamma^\circ, A^\circ, \neg B^\circ, \neg\Delta^\circ \vdash\end{array}}{\Gamma^\circ, \neg B^\circ, \neg\Delta^\circ \vdash \neg A^\circ}}{\Gamma^\circ, \neg\Delta^\circ \vdash (\neg B^\circ) \Rightarrow (\neg A^\circ)}}{\Gamma^\circ, \neg((\neg B^\circ) \Rightarrow (\neg A^\circ)), \neg\Delta^\circ \vdash}$$

Exercise 2.3.2 *Complete the translation of proofs treating the trivial case of left contraction and weakening and the case of $\wedge$, $\vee$ and $\neg$.*

The translation of proofs we have just exhibited actually proves the following

Theorem 2.3.3 (Soundness) *Let A be a classical propositional formula of LK. If A is provable in LK, then A° is provable in LJ.*

The converse is also true, but unlike the case of the embedding of LJ into LL, the proof is trivial.

Theorem 2.3.4 (Completeness) *Let A be a classical propositional formula of LK. If A° is provable in LJ, then A is provable in LK.*

Proof. A derivation of A° in LJ is also a derivation in LK, since LJ is a subsystem of LK. Now any formula A is classically equivalent to A°, and the result trivially follows. □

Remark 2.3.5 (Preservation of cut-free proofs) *Notice that a cut-free proof is mapped by the translation into a cut-free proof, as it is clear from the soundness proof for the embedding. Actually, we will be able to say much more than this: it is possible to define a notion of cut-elimination in LK that is preserved by the translation, in such a way that the cut-elimination theorem for LK can be inferred from the one of LJ. This uses in an essential way the fact that the translation respects the skeleton.*

Embedding LK into LL

If we now take the composition $(\_\circ)\star$ of the translations, we get back a translation from LK into LL that acts on formulae as follows:

$$(A \to B)^{\circ\star} = (\neg B^\circ \Rightarrow \neg A^\circ)^\star = !(\neg B^\circ)^\star \multimap !(\neg A^\circ)^\star = !?(B^{\circ\star})^\perp \multimap !?(A^{\circ\star})^\perp = ?!A^{\circ\star} \multimap ?!B^{\circ\star}$$

and translates sequents $\Gamma \vdash \Delta$ as $!\Gamma^{\circ\star} \vdash ?!\Delta^{\circ\star}$.
Again this is a plethoric translation: it is of course possible to obtain a more economical one (indeed, even two, one of them allows to recover then the economical intuitionistic translation; you can find all this in [DJS95c]).

CHAPTER 3

Proof Nets

citazione parallel syntax
Jean-Yves Girard
????

As is the case for classical or intuitionistic sequent calculus, a proof of a sequent $\Gamma \vdash \Delta$ (or $\vdash \Gamma^{\perp}, \Delta$ in our asymmetric presentation) can contain a lot of details that sometimes are uninteresting. For example, consider how many uninterestingly different ways there are to form a proof of $\vdash \Gamma, (A_1 \otimes A_2), \ldots, (A_{n-1} \otimes A_n)$ starting from a derivation of $\vdash \Gamma, A_1, A_2, \ldots, A_n$.
This unpleasant fact derives from the sequential nature of proofs in sequent calculus: if we want to apply a set S of n rules to different parts of a sequent, we cannot apply them in one step, even if they do not interfere with each other! We must *sequentialize* them, i.e. choose a linear order on S and apply the rules in n steps, according to this order. This phenomenon can lead us to $n!$ proofs that are intensionally different, but that we would like to consider as just one proof.
At this point, there is a natural question that needs to be answered: "is there a representation of proofs that abstracts from such uninteresting details?". A similar question is answered positively in the case of Intuitionistic Sequent Calculus by means of what is known, after Prawitz, as *Natural Deduction*.

3.1 Digression: Natural Deduction for ISC

Let's recall briefly the natural deduction presentation of the implication-only fragment of Intuitionistic Sequent Calculus, and some of its fundamental properties. For details, the interested reader ought to refer to [GLT90].

Definition 3.1.1 (Natural deduction)

- *A is a deduction of the formula A from hypothesis A*
- *if $\dfrac{D}{B}$ is a deduction of B from a multiset of hypothesis Γ, then*

$$\frac{\dfrac{D'}{B}}{A \Rightarrow B}$$

  is a deduction of $A \Rightarrow B$ from hypothesis Γ', where Γ' is Γ, eventually without some occurrences of A.

- *if $\dfrac{D_1}{A \Rightarrow B}$ is a deduction of $A \Rightarrow B$ from hypothesis Γ and $\dfrac{D_2}{A}$ is a deduction of A from hypothesis Γ', then*

$$\frac{\dfrac{D_1}{A \Rightarrow B} \quad \dfrac{D_2}{A}}{B}$$

is a deduction of B from hypothesis $\Gamma \uplus \Gamma'$, *where* $\uplus$ *is multiset union.*

It is possible to give a simple translation from any proof P in LJ to a natural deduction D, and to show that this translation is surjective (see[GLT90]), so that the two systems are equivalent as far as provability is concerned (i.e. proving a sequent $\Gamma \vdash A$ (recall that we are talking about Intuitionistic calculus) is equivalent to build a natural deduction of A with premisses taken (as much times as we want) from Γ). See [GLT90] for more details.
The natural deduction tree structure D associated to a proof P can be thinked of as the quotient of all the proofs that can be obtained from P by legal permutations of the rules in P.
The cut elimination property for ISC becomes a similar property still called cut elimination (but where a cut is defined in a different way) for Natural Deductions. A natural deduction tree can be put in exact correspondence with typed λ-calculus (that can be considered as just a notation for such deductions), and the cut-elimination in Natural Deduction corresponds exactly to β-reduction in λ-calculus.
In fact, we can present the simple typed λ-calculus in a way that mimicks directly the construction of a derivation tree.

Definition 3.1.2 (Simple typed λ-calculus)

- *x:A is a variable of type A*
- *if* $\dfrac{D}{M : B}$ *is a deduction of M:B from a multiset of hypotheses* Γ, *then*

$$\frac{\dfrac{D'}{M : B}}{(\lambda x : A.M) : A \Rightarrow B}$$

  is a deduction of $(\lambda x : A.M) : A \Rightarrow B$ *from hypothesis* Γ', *where* Γ' *is* Γ, *where all occurrences of x:A are discarded.*

- *if* $\dfrac{D_1}{M : A \Rightarrow B}$ *is a deduction of M:A⇒B from hypothesis* Γ *and* $\dfrac{D_2}{N : A}$ *is a deduction of N:A from hypothesis* Γ', *then*

$$\frac{\dfrac{D_1}{M : A \Rightarrow B} \quad \dfrac{D_2}{N : A}}{(MN) : B}$$

  is a deduction of (MN):B from hypothesis $\Gamma \uplus \Gamma'$, *where* $\uplus$ *is multiset union, if the declaration of types of common variables agree.*

From a simple typed λ-term it is possible to recover the full structure of the derivation of its well-typing. This is why such terms are also used as a "notation" for derivations in natural deduction.

Remark 3.1.3 *The rules for building a natural deduction tree have the following properties:*

- global well-formedness: the application of a building rule can require the inspection of the global tree already built (see the case for $\Rightarrow$).
- local correctness: if all the rules are applied properly (i.e. the proof tree is well-formed), then the deduction is correct.

Natural Deduction and Proof Structures

As already noted, a natural deduction is essentially a tree, with just one root that correspond to the just one conclusion in an intuitionistic sequent.
Can we adapt this construction to the case of Linear Logic? We have a first difficulty to face: sequents in LL, even in the symmetric presentation, have more than one conclusion, so that it seems not clear how to associate a *tree* to a class of derivations. Actually, it is possible to adapt natural deduction to this situation by using an *interpretation* of symmetric linear sequents in terms of intuitionistic linear sequents of ILL. Anyway, there is a more natural way (and much more interesting, as we will see later) to build a mathematical object that identifies uninterestingly different derivations in Linear Sequent Calculus.
We will consider no longer trees, but *graphs*, as suggested by the asymmetric presentation of LSC. For simplicity, the presentation that follows is restricted to the case of the multiplicative fragment MLL of linear logic.

Proof Structures. Concrete and abstract form.

Let's start by defining the notion of *sequential proof structure*, or *SPS*, by following exactly the construction of derivations in the asymmetric sequent calculus for MLL.

Definition 3.1.4 (sequential proof structure) *Define inductively "*S is a *SPS* with conclusions $X_1, \ldots, X_n$*", where* $X_1, \ldots, X_n$ *are multiplicative formulae, as follows:*

1. *ax link. For any multiplicative formula* A

$$\frac{}{A \quad A^{\perp}}$$

is a SPS *with conclusions* $A, A^{\perp}$.

2. *par link. If* S_1 *is a* SPS *with conclusions* $X_1, \ldots, X_n, A, B$, *then* S *is a* SPS *with conclusions* $X_1, \ldots, X_n, A \parr B$, *where* S *is :*

$$\dfrac{\begin{array}{c} S_1 \\ \vdots \\ A \quad B \end{array}}{A \parr B}$$

3. *tensor link . If* S_1 *is a* SPS *with conclusions* $X_1, \ldots, X_n, A$, *and* S_2 *is a* SPS *with conclusions* $Y_1, \ldots, Y_m, B$,

$$\dfrac{\begin{array}{cc} S_1 & S_2 \\ \vdots & \vdots \\ A & B \end{array}}{A \otimes B}$$

is a SPS *with conclusions* $X_1, \ldots, X_n, Y_1, \ldots, Y_m, A \otimes B$

4. *cut link . If* S_1 *is a* SPS *with conclusions* $X_1, \ldots, X_n, A$, *and* S_2 *is a* SPS *with conclusions* $Y_1, \ldots, Y_m, A^{\perp}$,

$$\dfrac{\begin{array}{cc} S_1 & S_2 \\ \vdots & \vdots \\ A & A^{\perp} \end{array}}{\qquad}$$

is a SPS *with conclusions* $X_1, \ldots, X_n, Y_1, \ldots, Y_m$

This definition allows to associate in a straightforward way an *SPS* to any derivation of an MLL sequent.

Exercise 3.1.5 (Embedding sequents into *SPS*'s) *Give a formal translation of a derivation of a sequent* $\vdash A_1, \ldots, A_n$ *into a* SPS *with conclusions* $A_1, \ldots, A_n$.

What is more interesting, we can prove that *any SPS* actually comes from a derivation of a sequent in *SPS*.

Theorem 3.1.6 (Correction of *SPS*'s) *If* S *is an* SPS *with conclusions* $A_1, \ldots, A_n$, *then there exists a derivation* $\mathcal{D}$ *in MLL of the sequent* $\vdash A_1, \ldots, A_n$. *Furthermore, the translation of this derivation is* S.

Proof. Exercise. □

Are these *SPS*'s the structures we were looking for in order to abstract from the too many details present in the derivations of the sequent calculus? Apparently yes: in a *SPS* there is no longer a trace of the order of application of non-interfering rules. Unfortunately, *SPS*'s enjoy the same properties of natural deduction, i.e. the *local correctness* and *global well-formedness*, and here to check the well-formedness of a tensor link in a *SPS* S we have to find a possible order of construction of S.
So, *SPS*'s are easy to build, but it seems to be very difficult to check that a structure looking like an *SPS* is actually an *SPS*, and not simply what is called a *concrete proof structure* (or *CPS*), that is a composition of links choosen among

$$\frac{}{A \quad A^{\perp}} \qquad \frac{A \quad B}{A \parr B} \qquad \frac{A \quad B}{A \otimes B} \qquad \underline{A \quad A^{\perp}}$$

and satisfying the conditions:

- every formula is conclusion of exactly one link
- every formula is premiss of at most one link.

Remark 3.1.7 (proof-structures) *In the literature,* CPS *are ususally called* pré-réseaux *or* proof structures*, while* SPS *are called* réseaux *or* proof-nets.

In the concrete form, the horizontal segments represent what are called *links*. Formulae directly above a link are called *premisses* of the link, while those directly below are called *conclusions* of the link. The formulae that are not premisses of any link in a *CPS* are called conclusions of that *CPS*. We also call *cut-formulae* those formulae that are premisses of a cut link, and *terminal formulae* of a *CPS* the cut-formulae and the conclusions of that *CPS*.

3.2 Correctness Criteria

We need to look for a criterion of correctness to ensure that a given *CPS* is actually an *SPS*, i.e. represents a derivation in MLL. We will provide here a rather detailed account of the most appealing criterions known to date (essentially due to Vincent Danos and Laurent Regnier, see [Reg92, Dan90]), and a quick survey of the other ones we know of. Essentially this will lead us to a tour of the following criteria:

Topological criteria

Long-trip [Gir87]

Contractibility [Dan90]

Acyclic-Connected (ACC) [Dan90]

Hereditary Secessiveness [Dan90]

Graph Parsing [GMM97]

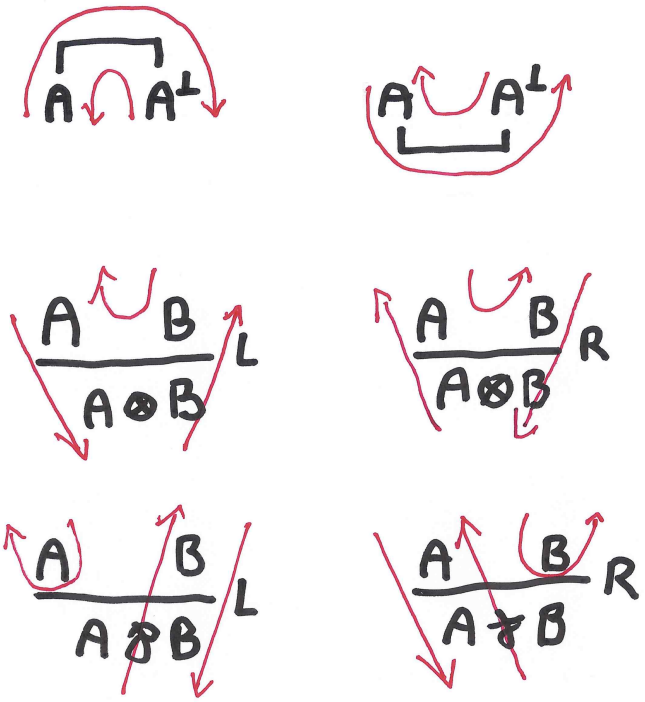
Deadlock-freeness [AD]

For each of them a proof of *soundness* and *completeness* can be found in the referenced papers. Here we mean by soundness the implication if a *CPS* is an *SPS* than it satisfies the criterion, and by completeness the reverse implication if a *CPS* satisfies the criterion, then it is an *SPS*. Typically, soundness can be shown in a straightforward way by induction on the structure of an *SPS*, while the completeness requires much more work. We will state all these criteria, but only study in detail two of them, namely Contractibility and Acyclic-connected, showing their soundness and completeness in a way that factors the difficult implication, and will give us two criteria for the price of one (this proof is an original elaboration of the original one in [Dan90]).

Topological Criteria

Let's start our survey by the *long-trip* criterion originally stated by Girard in his seminal paper. To do so, we look at each link of a proof-net as a router, having as ports the formulae that are premisses or conclusions of the link: to each link we associate a set of routing rules that tell us from which port we come out when we enter from a given port. Axiom and cut have a fixed behaviour, while tensor and par have two possible behaviours, determined by a local switch, that has two possible positions, L and R.

Definition 3.2.1 (Link routing tables and switches) *To each link in a* CPS *we associate a routing table, as depicted in the following:*



Tensor and par links can have two states, L and R, and their routing tables depend on these states.

The long-trip criterion can be stated as follows:

Theorem 3.2.2 (Long-trip) *A* CPS *is a* SPS *iff for every possible configuration of the switches associated to the par and tensor nodes one obtains a "long trip" (i.e. starting from any node one travels along a path that visits exactly once all formulae in the net in each direction).*

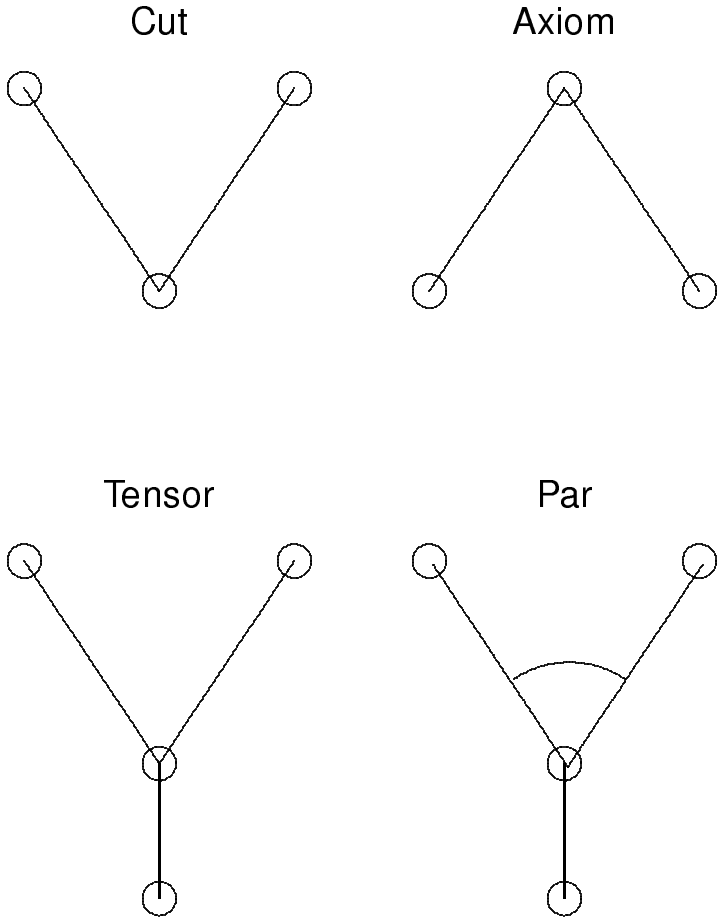
Towards ACC

This first criterion is of exponential complexity, so it is interesting to try and see if we can do better. Our first step will be to identify the underlying topological structure of a *CPS*: abstracting from the concrete details of the links of a *CPS*, we immediately notice that every link is essentially a connective with two (ax and cut link) or three (par and tensor links) connecting ports. The ax and cut connectives directly connect their two ports, while the par and tensor connectives connect two of the ports (the premisses) with a main port (the conclusion).

This observation suggests to represent abstractly each of these links by two connected segments in a graph.

Definition 3.2.3 *A* paired graph *is any undirected graph* $S = (E, V, C)$ *(where* E *is a set of* edges*, and* V *is a set of* vertices*), equipped with a set* $C(S)$ *of pairwise disjoint couples of co-incident edges*[1]*.The set of pairs* $C(S)$ *can also be seen as a binary relation on* $E \times E$*.*

Let's call simply *pairs* such couples of edges and *basis of a pair* a node common to the two edges, and let's say that two edges that belong to a same couple are *coupled*.
Graphically we denote the fact that two edges are coupled by a little arc tying them.
It is then straightforward to associate a paired graph S^- to a *CPS* S: just write an axiom link and a cut link as two edges, a tensor link as three edges and a par link as three edges, but where the two upper ones are *coupled* as in the following picture.



[1]Two edges are co-incident if they have at least one node in common.

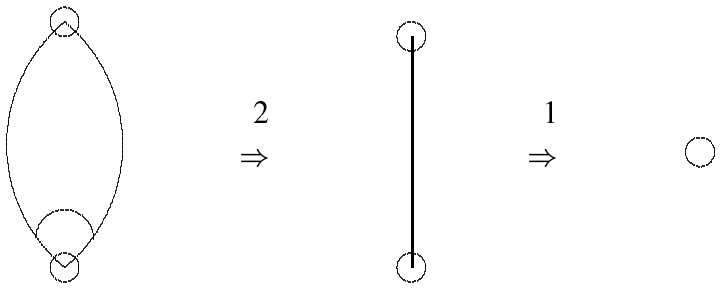
Now we can state the next correctness criterion:

Definition 3.2.4 (Acyclic-Connected) *Given a paired graph G, let $GC(G)$ be the set of graphs that can be obtained from it by removing exactly one edge for each pair of coupled edges.*
*We say that that G satisfies the Acyclic-Connected condition (*ACC*) if all the graphs in $GC(G)$ are acyclic and connected (i.e. they are trees). We will say that a* CPS *S satisfies* ACC *if S^- satisfies* ACC.

As we will see later, a *CPS* satisfying *ACC* is a *SPS*, but this criterion still invites to an exponential check: the graphs in $GC(S)$ are 2^p where p is the number of par link in the *CPS* S.

It is possible to do much better than that, using the following notion of graph reduction.

Definition 3.2.5 *We define the following graph rewriting rules on paired graphs:*



with the condition that:

- *rule 1 can be applied only to non-coupled edges connecting two* different *nodes*
- *rule 2 applies only to two coupled edges that connect the same two nodes.*

Exercise 3.2.6 *Show that the graph reduction just defined is strongly normalizing and weakly confluent (hence confluent).*

Definition 3.2.7 (Contractibility) *A* CPS *S is* contractible *iff S^- reduces to a single node.*

Again, we will see later that a contractible *CPS* is an *SPS*. A check for contractibility, though, can be performed in at most quadratic time.

Now it is very easy to show the sequence of implications:

$$S \text{ is a } SPS \;\Rightarrow\; S \text{ is contractible} \;\Rightarrow\; S \text{ satisfies } ACC.$$

The proofs are left as exercises.

Exercise 3.2.8 (*SPS*'s are contractibles) *Show by induction on definition 3.1.4 that a* SPS *is contractible.*

Exercise 3.2.9 (Contractible *CPS*'s satisfy *ACC*) *Let S be a paired graph reducing to S' by rule 1 or 2 above.*

- *Show that S satisfies* ACC *iff S' does.*
- *Infer from this that if S is contractible then S satisfies* ACC.

Completeness of the criteria

The two implications just proved tell us that both criteria are *sound*. Furthermore, we know by the second implication that *completeness* of the *ACC* criterion implies completeness of the contractibility criterion.
So we are left to show that a *CPS* satisfying *ACC* is indeed an *SPS*. To do so, we will again procede by induction on the structure of the graph, but a key point will be the possibility of splitting a given paired graph satisfying *ACC* into two smaller paired graphs that still satisfy *ACC*. Unlike what happens in the case of the proof of completeness of the original *long-trip* criterion, we will split the graph in correspondence of a *par* link, and not of a *tensor* link.

Definition 3.2.10 (splitting pair) *Let S be a paired graph. We say S is* split *if there is a pair p of S s. t. the basis of p is disconnected from the other two extremities of p in $S \setminus p$. Such pair is called* splitting.

Exercise 3.2.11 (splitting preserves *ACC*) *Let S be a paired graph split by p; show that if S satisfies* ACC *then $S \setminus p$ has two connected components, also satisfying* ACC. *Show the converse.*

Is it always possible to find a splitting pair in a paired graph that satisfies ACC? Yes, it is! But here we find the only hard part of the proof.

Theorem 3.2.12 (Paired graphs with at least one pair are split) *Let S be a* finite *paired graph satisfying* ACC. *If S counts at least one pair then it is* split.

Proof. This is a difficult theorem: see [Dan90], Proposition 4-5, for details. See also the Appendix. □

Exercise 3.2.13 *Show a counter example to the theorem above in case one allows infinite graphs.*

Now we have all the necessary tools to prove the completeness of the *ACC* criterion, and hence of the Contractibility criterion too.

Theorem 3.2.14 (Completeness) *Any* CPS S *satisfying* ACC *is an* SPS.

Proof. We will procede by induction on the size of S, but the inductive steps will force us to look at S not just as a *CPS* or a *SPS*, but as a slightly more general structure.

Exercise 3.2.15 *Add to definition 3.1.4 the following clause:*

1'. hypothesis-link. For any multiplicative formula A

$$\overline{A}$$

is a SPS *with hypotheses with conclusions A.*

to have the definition of a SPS *with hypotheses. Similarly extend the definition of* CPS *to cope with hypotheses (this is only induction loading).*
Now, let S be a CPS *w. h. satisfying* ACC*:*

- *if S has no pairs show by induction on the number of $\otimes$ that S is a* SPS*w.h.,*
- *if S has a pair use the non obvious graph-theoretic property above to show by induction on the number of $\parr$ that S is a* SPS*w.h..*

□

Exercise 3.2.16 (Danos-Schellinx) *Let A be a* provable *multiplicative formula.*

1. *Suppose S is a cut-free proof-net proving A, so it consists in the "tree" of A, say $T(A)$, together with a set of axiom links over its leaves (the atoms of A). Show that $C(A) = N(A) + 1$, where $C(A)$ is the number of connected components of $T(A)$ under any switching s, and $N(A)$ is the number of axiom links in S.*
2. *Prove that $P(A) = N(A)$, where $P(A)$ is the number of $\parr$ connectives in A*
3. *infer from this that $\vdash A, A, \cdots, A$; and better that if A_1, A_2, …, A_n are provable (for $n > 1$), then $\vdash A_1, A_2, \cdots, A_n$*
4. *again infer from point 1 that if $P(A) + N(A) \equiv P(B) + N(B)$ (mod 2), then $\vdash A, B$.*

Graph Parsing criteria

Once we have seen how correctness can be established easily via contractibility of the paired graph associated to a *CPS*, it is not difficult to derive a criterion working directly on the proof structures, and that has the same behaviour [Laf95]: this is done by defining a graph parsing grammar that will rewrite a correct *CPS* having conclusion Γ to a single special node NET having the same conclusions. Here we present the grammar corresponding to MLL taken from [GM97], where it has been extended to a version of full MELL where weakening is only allowed on the axioms.

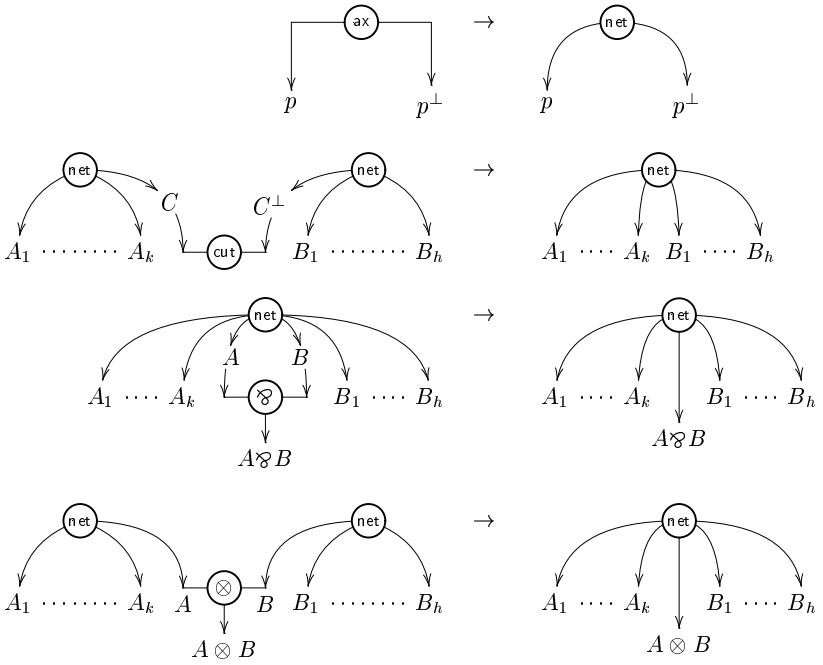


Table 3.1: Graph parsing rules for MLL

Theorem 3.2.17 (Graph Parsing [GM97]) *A* CPS *with conclusions* Γ *is a* SPS *iff it rewrites to the node* NET *with conclusions* Γ *under the graph grammar in table 3.1.*

Remark 3.2.18 *The correctness proof proceeds exactly like the one for the Contractibility criterion, i.e. via a reduction to ACC. Unlike the case of the two reduction rules in the contractibility criterion, this graph rewriting system is not confluent in general, but this is not problematic since one can show that if a reduction path recognizes the net as correct, then all paths do, and this is enough.*

Handling the MIX rule

In the proof of correctness for the ACC criterion, the only difficult part really uses (as you can see in the Appendix), a weaker property, called Weak-ACC or just AC, that asks the correction graphs to be acyclic, but not necessarily connected. This allows to capture CPS that are *families* of SPS's, and correspond to proof in MLL extended with the following MIX rule:

$$\frac{\vdash \Gamma \qquad \vdash \Gamma'}{\vdash \Gamma, \Gamma'} MIX$$

The MIX rule allows for example to prove the linear implication

$$A \otimes B \multimap A \parr B$$

It is then a simple exercise to adapt the definitions and proofs just seen to obtain the following

Theorem 3.2.19 (AC criterion) *A CPS satisfying AC is a family of SPS's, and represent a proof in MLL+ MIX.*

Deadlock freeness, MIX and ACC

A final criterion that we want to present here is due to Andrea Asperti, and tries to highlight a sort of relation with concurrent processes: here formulae are processes, and connectives are used to build new processes out of existing ones, in the process-algebras tradition, according to the following informal intuition

$A \otimes B$ is the interleaving of process A and process B

$A \parr B$ is process A run in parallel with process B

$A^{\perp}$ is the process whose only action is to synchronize with A

If we take this point of view, a *CPS* becomes the representation of a network of processes interconnected via the axiom and cut links, and it is natural to ask questions about the dynamics of this parallel system, i.e. what happens when a process A is activated (we will write $\uparrow A$ to denote activation) and when does it terminate (we will write $A\downarrow$ to denote termination).
According to the intuition given above, the natural evolution rules of the system are the following:

Definition 3.2.20 (Activation and termination of processes) *Activation and termination are propagated along the links according to the following rules:*

- *evolution of an axiom link:*

$$\uparrow\overline{A \;\; \uparrow A^{\perp}} \quad \rightarrow \quad \overline{A\downarrow \;\; A^{\perp}\downarrow}$$

- *evolution of a par link:*

$$\frac{A \quad B}{\uparrow A \parr B} \quad \rightarrow \quad \uparrow\frac{A \quad \uparrow B}{A \parr B} \qquad\qquad \frac{A\downarrow \;\; B\downarrow}{A \parr B} \quad \rightarrow \quad \frac{A \quad B}{A \parr B\downarrow}$$

- *evolution of a tensor link: to every tensor is associated a switch with two possible states L and R (like in Girard's criterion), and the evolution of the link depends on this state*

$$\frac{A \quad B}{\uparrow A \otimes B}\,L \quad \rightarrow \quad \frac{\uparrow A \quad B}{A \parr B}\,L \qquad\qquad \frac{A \quad B}{\uparrow A \otimes B}\,R \quad \rightarrow \quad \frac{A \;\; \uparrow B}{A \parr B}\,R$$

$$\frac{A\downarrow \;\; B}{A \otimes B}\,L \quad \rightarrow \quad \frac{A \;\; \uparrow B}{A \parr B}\,L \qquad\qquad \frac{A \quad B\downarrow}{A \otimes B}\,R \quad \rightarrow \quad \frac{\uparrow A \quad B}{A \parr B}\,R$$

$$\frac{A \quad B\downarrow}{A \otimes B}\,L \quad \rightarrow \quad \frac{A \quad B}{A \parr B\downarrow}\,L \qquad\qquad \frac{A\downarrow \;\; B}{A \otimes B}\,R \quad \rightarrow \quad \frac{A \quad B}{A \parr B\downarrow}\,R$$

- *evolution of a cut link: in a process net all cut links between a formula A and its dual $A^{\perp}$ are handled* exactly *as if they were replaced by a tensor link with conclusion $A \otimes A^{\perp}$ (the same can be done in Girard's criterion)*

In this view, a *deadlock* is a configuration that can no longer evolve, yet contains at least one active processes. It turns out that a *CPS* which is correct for MLL with the MIX rule is precisely a process structure that does not produce deadlocks during the evolution of the system.

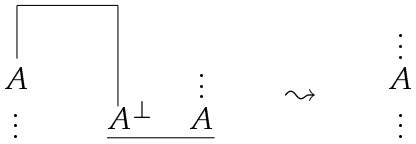
Theorem 3.2.21 (An *SPS* with MIX is a deadlock-free *CPS*) *A* CPS *is an* SPS *for MLL with the MIX rule iff under any configuration of the tensor switches the resulting process net is deadlock free (i.e. after activating all terminal formulae of the net one can reach the final state where all terminal formulae have terminated).*

This treatment can be extended to reacher process algebras: for details see [Asp91].

Remark 3.2.22 *It is still interesting to remark that, while a direct proof of completeness is possible, a very easy proof can be given by reducing the criterion to the Weak-ACC one [AD93]. Again, we see here that the interest of the ACC criterion is not its complexity, still exponential, but its versatility in providing proofs of completeness by reduction to it.*

3.3 Reduction of proof-nets.

Now that we know how to spot a real proof net among the many *CPS*, the main question is: can we perform cut elimination directly on the proof nets? Is it easier than in the sequent calculus? If so, how do we prove that such process preserves correctness?
In the MLL case, the answer is a definite yes, and everything is very simple: one can reduce a proof-net either this way (axiom move):



or that way (multiplicative move):

These reduction preserve correctness, are confluent and strongly normalizing, so provide an effective way of performing cut elimination that is way simpler than what can be done in the sequent calculus directly: thanks to the parallel syntax provided by the proof nets, we have no need whatsoever to go through awkward commutation lemmas here.

Exercise 3.3.1 (Reduction preserves correctness)

1. *Show that the axiom-move preserves correctness.*
2. *Call R the lhs of the multiplicative-move; take S to be any switch of R and call S' the graph obtained from S by erasing the three links $\otimes$, $\parr$ and cut concerned by the move. Show that among A, B, $A^\perp$ and $B^\perp$ the only two connected vertices are $A^\perp$ and $B^\perp$.*
3. *Deduce from the previous point that the multiplicative move also preserves correctness.*

So these two moves preserve correctness, and the right handside is indeed a proof-net if the left hand side is one.

Exercise 3.3.2 (Confluence and Strong Normalisation)

1. *Show that this rewriting process is noetherian;*

2. *show it is confluent (by the previous point it is enough to prove that it is locally confluent) and every reduction to normal form have the same length.*

Exercise 3.3.3 (splitting $\otimes$) *Let S be a* CPS*, a $\otimes$ in S is said to split it if its erasure splits S in two connected components.*

1. *show that in a* CPS *satisfying* ACC*, with no terminal $\parr$-links, there is such a $\otimes$ (unless the structure is an axiom link).*

2. *(Andreoli-Pareschi) show that under the same hypotheses one can even find an hereditarily splitting $\otimes$, i.e., a splitting $\otimes$, the premises of which are $\parr$'s or again hereditarily splitting $\otimes$.*

Note that there is a hidden assumption in the definition of the axiom move. Namely that both A's are different occurrences of A. If they were the same, then the situation would be the 'black-hole':

$$\frac{\overline{A \quad A^{\perp}}}{\overline{A \quad A^{\perp}}},$$

forever rewriting to itself. Of course this black hole is not a proof-net. So there is more about acyclicity than sequentialisation, it also prevents the formation of looping configurations.
Bechet [**?**] recently proved a kind of converse to this remark that makes a real happy end to the story of correctness. Suppose a proof-structure S is cyclic, then there are proof-nets $R_1, \ldots, R_n$ such that S cut on $R_1, \ldots, R_n$ produces a black hole. Thus acyclic proof-structures are *exactly* those that will never produce any such configuration, whatever the context in which they are plugged.

3.4 Proof Nets for MELL

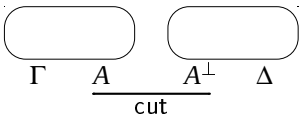
We have seen up to now what are the proof nets for MLL, whose behaviour is fully satisfactory. For MELL, we can proceed to a similar construction, but the traditional presentation of exponentials, as we will see, is less satisfactory. Here follows the traditional presentation of Proof Nets for MELL:

Definition 3.4.1 *The set of proof nets, denoted by PN, is the smallest set satisfying the following properties:*

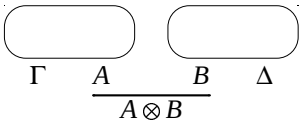
- *An A-axiom link is a proof net with output formulae $A, A^{\perp}$, written as*

$$\frac{}{A \quad A^{\perp}}\,\mathsf{ax}$$

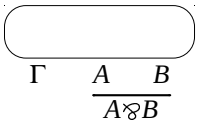
- *If S is a proof net with output formulae A, Γ and T is a proof net with output formulae $A^{\perp}, \Delta$, then adding an A-cut link between A and $A^{\perp}$ yields a proof net with output formulae Γ, Δ written as*



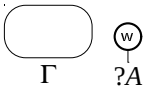
- *If S is a proof net with output formulae A, Γ and T is a proof net with output formulae B, Δ, then adding an A-B-times link between A and B yields a proof net with output formulae $\Gamma, A \otimes B, \Delta$ written as*



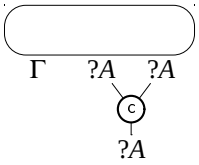
- *If S is a proof net with output formulae Γ, A, B, then adding an A-B-par link between A and B yields a proof net with output formulae $\Gamma, A \parr B$ written as*



- *If S is a proof net with output formulae Γ, then adding an A-weakening link yields a proof net with output formulae $\Gamma, ?A$ written as*

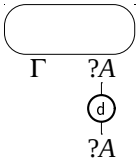


- *If S is a proof net with output formulae $\Gamma, ?A, ?A$, then adding an A-n-contraction link yields a proof net with output formulae $\Gamma, ?A$ written as*

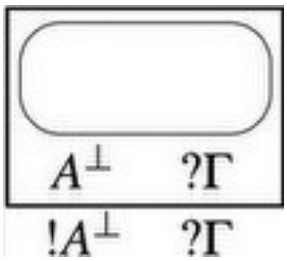


We say that $?A, \ldots, ?A$ *are the inputs of the contraction node, and the conclusion formula* $?A$ *is the output of the contraction node.*

- *If* S *is a proof net with output formulae* Γ, A*, then adding an* A*-dereliction link yields a proof net with output formulae* $\Gamma, ?A$ *written as*

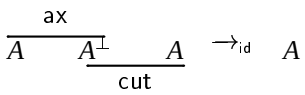


- *If* S *is a proof net with output formulae* $?\Gamma, A$*, then adding an* A*-*$?\Gamma$*-box link yields a proof net with output formulae* $?\Gamma, !A$ *written as*

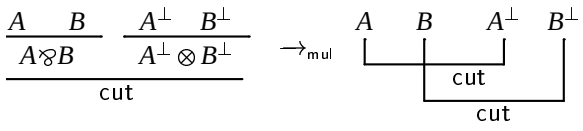


Definition 3.4.2 *The cut-elimination procedure over* PN *is performed via the rewrite relation defined as follows:*

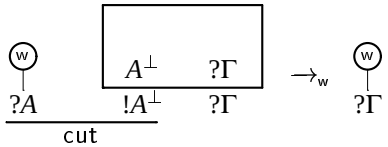
- *axiom:*



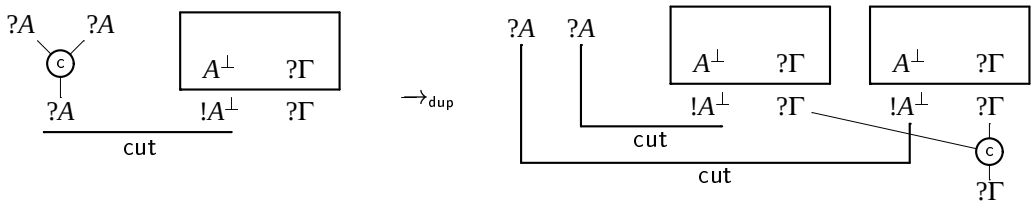
- *times-par:*



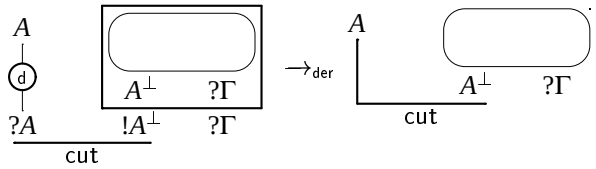
- *weakening-box:*



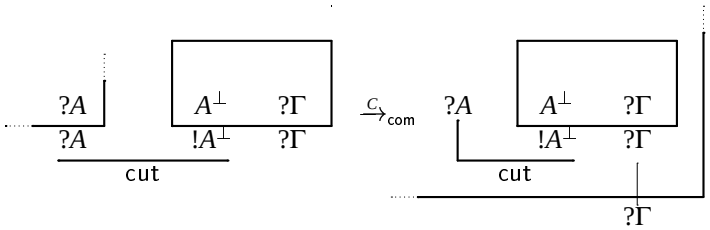
- *contraction-box:*



- *dereliction-box:*



- *box-box:*



For this system one can show the same results as for MLL, but in a less straightforward way.

Theorem 3.4.3 (Cut-elimination in MELL) *The reduction rules are confluent, strongly normalizing and preserve correctness.*

For a proof, the interested reader is referred to the seminal paper [Gir87].

Remark 3.4.4 *This original presentation of proof nets for full MELL is not totally satisfactory for several reasons:*

- *binary contraction nodes mimick exactly in the net the order of application of contraction rules in the sequent derivations, so we have lost the parallelisation of syntax: for each of the exponentially many ways to derive* $\vdash A$ *from* $\vdash \underbrace{A, \ldots, A}_{n}$*, we get a different tree of contraction nodes in the*

corresponding net. Also, the order of application of the contraction and promotion rules is still visible in the net. One can remedy to these defect by working in nets up to associativity of contraction nodes and traversal of box borders for contraction nodes. This has been done by several authors: to give a cleaner presentation of the Geometry of Interaction equations for the semantics of λ-calculus in [DR95], where reduction is defined to maintain specific normal forms up to these rules, and to give a tight simulation of the reduction used for λ-calculi with explicit substitutions in [DCK96], where more liberal reductions are used.

- *due to the presence of boxes, we lost the nice local reduction property that was so attractive in the proof nets for MLL. As we will see later, one can get locality of reduction back by different means, all leading to various flavours of sharing graphs that are used to implement optimal reduction.*

CHAPTER 4

Computational aspects of the Embeddings into Linear Logic

algebrization du calcul
Jean-Yves Girard
????

The embeddings from LJ into LL naturally suggest an embedding of natural deduction intuitionistic proofs (that correspond to types λ-calculus), into proof-nets. In this chapter we will show how the more refined structure of proof-nets allows to perform a finer analysis of sequential computation, in at least the following three possible ways:

local and optimal reduction the reduction in MLL proof nets is local, as opposed to the global substitution or grafting mechanism which is the only available in intuitionistic natural deduction. This locality of the reduction, which still can be mantained in MELL at the price of some more graph combinators, allows to (sort of) implement Levy's optimal reduction strategy for the λ-calculus.

algebrization of the reduction instead of reducing a proof net, one can try to expose the essence of such a reduction, by introducing an algebraic description of the paths that make up a proof net, with the axioms that insure a correct composition. A proof net is then decomposed in a family of paths, out of which one can, by algebraic manipulations, recover its normal form. This plan forms the core of Girard's Geometry of Interaction.

game semantics finally, another way to compute with proof nets is by interpreting linear formulas as suitable games and linear derivations as strategies in these games. Similarly to what happens with paths, strategies can be composed and one can then obtain the strategy associated to the normal form by formal manipulation.

Here too, we will try to be faithful to the spirit of the book: we focus on a simple case (the linear lambda calculus and MLL), where a detailed exposition will allow any reader to become familiar with the fundamental ideas, and then refer to the appropriate literature for the case of the full λ-calculus (but the next chapter will help in bridging the gap between MLL and MELL by studying all these issues in a powerful fragment of LL known as Elementary Linear Logic).

4.1 Embedding λ-calculus in Linear Logic

A linear λ-term is a term of which each variable occurs exactly once. Such terms are simply typable in the type-assigment system we introduced in 3.1.2, as stated in the following

Exercise 4.1.1 (Simple types for linear terms) *Show that any linear term is simply typeable.*

Proof. (Hint: prove first the property for terms in β normal form, and then show that a linear term is simply typeable if and only if one of its β-contractums is.) □

For linear lambda terms, both embeddings we have seen (the plethoric and the economical one) can be simplified by removing all exponentials (this is indeed the key to this simple case: neither duplication nor erasure is allowed).

Definition 4.1.2 (Simple embedding from LJ to MLL) *Define* $(A \to B)^\star$ *to be* $A^{\star\perp} \parr B^\star$ *and if* A *is atomic* $A^\star$ *to be* A.

Now, to any (simply typeable) term t we can associate an unique intuitionistic derivation $D(t)$ in natural deduction, and then to this derivation we can associate a (no longer unique, but one can choose the natural one) derivation $S(D(t))$ in the sequent calculus LJ. This gives, via $\_^\star$, a derivation $S(D(t))^\star$ in MLL, which we know how to map into proof-nets. This process provides us, in general, with an embedding of λ-terms into proof-nets, that depends on the chosen translation from LJ into LL.

In our simple case, we can make the translation explicit: the principle of the translation is that $M^A[x_1^{B_1} \dots x_n^{B_n}]$ is turned into a proof-net $M^\star$ with conclusions $B_1^{\perp\star}, \dots, B_n^{\perp\star}$ and $A^\star$.

Definition 4.1.3 (Embedding linear lambda terms in MLL proof nets) *The translation is defined by induction on the derivation as follows (dropping the* $\star$*'s over the formulae in the net)*

- x^A *is turned into an axiom link between* $A^\perp$ *and* A*;*
- *let* $M^{A\to B}$ *and* N^A *be two terms, then* $(M)N$ *is turned into:*

$$\dfrac{M^\star \quad \dfrac{\begin{array}{c}N^\star\\ \vdots\\ A\end{array} \quad B^\perp}{A \otimes B^\perp}}{A \to B} \qquad B$$

- *Let* M^B *be a term and* x^A *a free variable in* M*,* $\lambda x\ M$ *is turned into:*

$$\dfrac{\begin{array}{c}M^\star\\ \vdots\\ B\end{array}}{A \to B}$$

What is wrong with this translation is that normal terms are not turned into cut-free proof-nets. Indeed, one can devise a better translation where redexes of M are one-to-onely mapped to cuts of $M^\star$, this map preserving types. Moreover this translation induces an interesting equivalence relation between λ-terms (see [Reg91, Dan90] for details). But let us content ourselves with this simple encoding.

Now one can see proof-nets as a relaxed way to write down λ-terms. If one remembers the reduction moves for proof-nets, then the benefit of such an embedding is clear: we now have local reductions, whereas at first sight β-reduction even on linear terms did not seem to be local. Surely this is a very satisfying representation of terms, alas, we are expelled from this girardian paradise as soon as we step out from linear terms, for erasure and duplication deprive us from this locality of reductions in the general case ... but still we can do better.

4.2 Local Reduction

The embedding just defined allows us to *implement* the β-reduction rule

$$(\lambda x.M)N \to_\beta M[N/x]$$

locally in MLL proof nets precisely via the reduction moves in the proof nets. Actually, we can forget about the formulae and just remember the links, thus getting closer to the usual graph reduction.

For example, here is the implementation of the reduction $(\lambda x : A.x)y \to_\beta y$ via the local reductions inside proof nets:

The simplistic framework of the linear lambda calculus does not allow to expose much of what is done in this field, because the real difficulty is to handle sharing, and no sharing is necessary for linear lambda terms. Let us just say for the moment that the locality of the reduction can be mantained in MELL (which is the subsystem of LL we need to capture the full λ-calculus) at the price of some more graph combinators, and allows to (sort of) implement Levy's optimal reduction strategy for the λ-calculus. We say "sort of", here, because what really has been done is to provide a graph implementation where just the number of graph reduction steps corresponding to *initiating* a β reduction is optimal, while the number of graph reduction steps necessary to bring the β-reduction to *completion* is in general difficult to evaluate, and here is where all the work on optimal reduction takes place: without this explanation of the misleading terminology which is commonplace in the field, a reader could be much surprised by the many papers dealing with "optimizations" of "optimal" reduction.

We will present in detail the optimal reduction rules for a more powerful fragment of LL in the next chapter.

4.3 Dynamic Algebra

Encoding proof-nets in the dynamic algebra.

Let us study another way to store the information which hides in a proof-net and is really needed for computation. Take a proof-net without cuts (also called a pure net looking like:

$$\dfrac{\dfrac{\vdots}{A} \quad \dfrac{\dfrac{\vdots}{B} \quad \boxed{C^{\perp}}}{B \otimes C^{\perp}}}{A \parr (B \otimes C^{\perp})} \qquad \dfrac{C \quad \dfrac{\vdots}{D}}{C \otimes D}$$

a *section* along an axiom link in this proof-net is a path α like:

$$C \otimes D, C, C^{\perp}, B \otimes C^{\perp}, A \parr (B \otimes C^{\perp})$$

linking two roots of the pure-net. Suppose now L and R to mean respectively left and right, and $\star$ to mean upwards, then the *weight* of α:

$$w(\alpha) = R^2 L^{\star}$$

is enough to describe it, if one knows which roots α is connecting (beware composition is to the left). Therefore, formula-trees and axiom links of a pure-net P can be represented (up to the identification of $\otimes$ and $\parr$, but the distinction only matters for correctness, see again the multiplicative move) by a matrix $M = (M_{AB})$:

- whose entries are the roots of P;
- where $M_{AB} = \sum w(\alpha)$, the sum extending over all sections connecting A to B.

In case the net we are given is not pure, we will nevertheless decompose it in its pure part, to which we will associate a matrix M as above, and in a cut-link part, to which we associate a matrix σ:

- with the same entries as M;
- where $\sigma_{AB} = 1$ iff A and B are cut-linked and 0 otherwise.

Remark 4.3.1 *The idea behind this construction is that what really counts in a net is just the set of paths connecting each final formula of the net to each other formula, and such paths can be decomposed into paths along a section followed by a cut-link, followed by another path along a section and so on. L and R (for left and right) are like Pollicino's white stones keeping track of the path, augmented with an involutive operation $\_^\star$ to remember if we are going up or down along a section, L being down left and $L^\star$ being up left.*

The set in which these coefficients are equipped with an *associative* product that is *distributive* over the sum, together with some reasonable equations like an involutive $\star$, is called the *dynamic algebra*. More formally

Definition 4.3.2 (The dynamic algebra Λ^* (for MLL)) *The dynamic algebra is a semiring generated by $\{L, R\}$, with an involutive operation $\_^*$ defined as*

$$\begin{array}{lcl} u^{**} & = & u \\ 0^* & = & 0 \\ 1^* & = & 1 \\ (u+v)^* & = & u^* + v^* \\ (uv)^* & = & v^* u^* \end{array}$$

and satisfying the following consistency axioms:

$$\begin{array}{lcl} L^*L & = & 1 \\ R^*R & = & 1 \end{array} \qquad \begin{array}{lcl} L^*R & = & 0 \\ R^*L & = & 0 \end{array}$$

Remark 4.3.3 *In the algebra, 1 models an empty path, hence the identity over paths, while 0 models an* illegal *path, that composed with any other path stays illegal, hence the multiplicative zero over paths.*
The consistency axioms also have a very intuitive reading: the first two just say that going up and then coming back down bring us precisely to where we came from; the second two are there to say that it is illegal to go down left and then up right or down right and the up left: indeed, this combination only can happen if we are crossing a cut link and we do not take the right path, that connects left premisses of a $\otimes$ (resp. $\parr$) to left premiss of a $\parr$ (resp. $\otimes$), and does not *mix right with left.*

Computing the paths in a net N with the two matrices introduced above amounts to compose sections along cut-links, i.e., to build some path in N taking alternatively a section and a cut, which we weight with the product of the weights of

the sections it uses. To put this in a formula:

$$\mathrm{EX}(\sigma, M) = P(M + M\sigma M + M\sigma M\sigma M + \cdots)P,$$

where P is the projection on those roots which are not cut-linked to any other root, which can be easily seen to be $(1 - \sigma^2)$.
Now one can prove:

Theorem 4.3.4 *$EX(\sigma, M)$ is an invariant of reduction moves, and $(\sigma M)^{2^{|N|+1}} = 0$ where $|N|$ is the number of axiom moves it costs to reach the normal form of N.*

Example 4.3.5 All this is best understood by looking at a worked example: let's consider the translation of the linear λ-term $(\lambda x : A.x)y$ where y is a free variable of type A. According to our definition of the translation, we get the following proof net: x which reduces, in 3 steps of cut elimination, to the identity net

$$\overline{A \quad A^{\perp}}$$

Let's give a number to each of the conclusions of the net cosidered without the cut links: we have 4 formulae, $F_1 = A^{\perp} \parr A, F_2 = A \otimes A^{\perp}, F_3 = A^{\perp}, F_4 = A$. Now we can write down the two matrices M and σ, indexed over $F_1, \ldots, F_4$.

$$\sigma = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad M = \begin{pmatrix} RL^* + LR^* & 0 & 0 & 0 \\ 0 & 0 & L^* & R^* \\ 0 & L & 0 & L^*R = 0 \\ 0 & R & R^*L = 0 & 0 \end{pmatrix}$$

Also, the matrix $1 - \sigma^2$ is just the projector necessary to keep only the paths between the conclusions of the net that are not cut with any other formula:

$$1 - \sigma^2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

You can check that $(\sigma M)^3 = 0$ and that $\mathrm{EX}(\sigma, M)$ is

$$\mathrm{EX}(\sigma, M) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

which corresponds exactly to the identity net over $A^{\perp}$ and A, which in turn is the translation of a free variable y of type A. □

Remark 4.3.6 *Notice that this perfect match between β-reduction and execution formula is commonplace only over linear terms: as soon as we allow multiple occurrences of a same variable, duplication and erasing lead us to handling* sharing *of data in the proof nets. It can well be the case, then, that if a term t reduces to t', the nets associated to t and t' present different degrees of sharing.*

Definition 4.3.7 (Regular path, maximal path) *A path in a proof net (i.e. a monomial of the algebra) is called* maximal *if it cannot be extended, and* regular *if its weight is not null.*

Exercise 4.3.8 *Let r be any of the two reduction move and P a proof-net.*

1. *Build a one-one correspondence from maximal regular paths of P onto maximal regular paths of $r(P)$,*

2. *deduce from the previous point the theorem above.*

Exercise 4.3.9 (η-reduction) *Translate $\lambda y\,(x)y$ into pure-nets as shown in 4.1.3; reduce it, then compare with the translation of x and define η-reduction in proof-nets.*

Exercise 4.3.10 (η-invariance) *1. Show that the degree of nilpotency of P is invariant under η-reduction (see exercise 4.3.9 for the definition).*
2. Suppose M is a linear λ-term and M' an η-expansion of it such that M' is β equivalent to M. Show that the reduction length (reduction length is independent of the strategy for linear terms; check !) is greater for M' than for M. Conclusion: the degree of nilpotency does not measure reduction lengths.

Exercise 4.3.11 (regular extensions) *Say a monomial m in the dynamic algebra is* stable *if it has the form $m_1 m_2^*$ where m_1 and m_2 are monomials without any $\star$'s (the void product means 1). Note that sections have normal weights.*
1. Show that for all monomials m, either $m = 0$, or there is a stable m' s.t. $m = m'$.
2. Infer from this that every regular non maximal path in a proof-net admits a regular extension.

Exercise 4.3.12 (a model) *Consider the set $PI(X)$ of partial injections (short, pi) from a countable set X into itself. Define in this set the involution and the product of the dynamic algebra, and find two pi's interpreting L and R.*

4.4 Taming the full typed λ-calculus

It is possible to extend this treatment of computation to the full typed λ-calculus, as we will briefly hint here, and event ot the pure untyped λ-calculus (see [Dan90, Reg92] for details). The first step is to define a translation suitable for non-linear terms, so we need exponentials, and we have many choices, but the easier to present is a transaltion associated to the economical embedding of LJ into LL:

Definition 4.4.1 (From typed terms to MELL proof nets) *The translation of a type is the same as for the economical embedding:*

$$\begin{array}{lcll} A^* & = & A & \text{if } A \text{ is atomic} \\ (A\ B)^* & = & ?((A^*)^\perp) \parr B^* & \end{array}$$

and the translation of a derivation is given by induction on the typing derivation as follows:

- *If* $\Gamma, x : A \vdash x : A$ *is an axiom, with* $\Gamma = C_1, \cdots, C_n$*, then its translation is:*

- *If* $\Gamma \vdash \lambda x : B.M : B\ C$*, with* $\Gamma, x : B \vdash M : C$*, then its translation is*

- *If* $\Gamma \vdash (M\ N) : A$*, with* $\Gamma \vdash M : B\ A$ *and* $\Gamma \vdash N : B$*, then we translate* $(M\ N)$ *as:*

As for the dynamic algebra, one needs to extend it with some additional generators and operators necessary to code the new form that a section can assume once we allow exponentials in the nets: four new constants r, s, t, d code the right or left choice in a path traversing a contraction node, the auxiliary ports of a box and the crossing of a dereliction operator respectively, while a new unary operator ! marks a path crossing the main door of a box.

The following new axioms take care of the identification and annihlation of paths along a reduction of a proof net, and complete the definition of the dynamic

algebra for MELL, once added to the ones for MLL that we have seen above:

$$(!(x))^* = !((x)^*)$$
$$!(x)!(y) = !(xy)$$

Annihilation
$$r^*r = s^*s = d^*d = t^*t = 1$$
$$s^*r = r^*s = 0$$

Commutation
$$!(x)r = r!(x)$$
$$!(x)s = s!(x)$$
$$!(x)t = t!!(x)$$
$$!(x)d = dx$$

Remark 4.4.2 (Preservation of paths under cut-elimination) *In this richer setting, we no longer have the simple correspondence between equations in the algebra and cut-elimination that holds in the MLL case: here a path in a net R before cut-elimination is not equal in general in the algebra to the residual of that path in the proof net R' obtained after one step of cut-elimination. What counts is the preservation property: proper (non-zero) paths in R will survive the process of cut elimination and stay non-zero under the action of the dynamic algebra. This unpleasant phaenomenon is quite related to the accumulation of brackets and croissants in sharing graphs* à la *Lamping [**?**].*

Remark 4.4.3 (Categorical analogies) *Note that the equations associated to d and t remind of a monad or a comonad. Indeed, $!$ can be seen as a comonad in a suitable category of proofs (show that $\vdash !A \multimap A$ and $!A \multimap !!A$). More details can be found in [Asp93].*

Remark 4.4.4 (Simpler dynamic algebras) *If one works with n-ary contraction nodes, it is possible to give a much more compact presentation of the dynamic algebra, as done in [DR95], to which we refer the interested reader for more details.*

This finishes our exposition of the Geometry of Interaction, but does not exhausts the subject: what can be done for example for the additives? Are there different presentation of the dynamic algebra? These and many other questions are the object of active ongoing research at the time of writing. The interested reader can find some glimpse of this work in papers like [AJ94, Gir95]

CHAPTER 5

Conclusions and annotated bibliography

Here we add a small selection of annotated research and exposition papers in different fields of application of linear logic, while referring the interested reader to up-to-date on-line WEB bibliographies for a comprehensive index of research in the domain. Our intent is to provide a few pointers to paper that are significant in view of the goal of this book: namely, provide a broad view of relevant applications and elementary concepts.

Index

Bibliography

[AD] Andrea Asperti and Giovanna Dore. Yet another correctness criterion for MLL with MIX. Available as `ftp://ftp.cs.unibo.it/pub/asperti/ADlfcs.ps.Z`.

[AD93] Andrea Asperti and Giovanna Dore. Yet another correctness criterion for multiplicative linear logic with MIX. Manuscript, November 1993.

[AJ94] Samson Abramsky and Radha Jagadeesan. New foundations for the geometry of interaction. *Information and Computation*, 111(1):53–119, 1994. Extended version of the paper published in the Proceedings of LICS'92.

[Asp91] Andrea Asperti. A linguistic approach to deadlock. TR LIENS 91-15, Ecole Normale Superieure. DMI, 1991.

[Asp93] Andrea Asperti. Linear logic, comonads and optimal reductions. Manuscript, to appear in Fundamenta Informaticae, 1994, special issue devoted to Categories in Computer Science, September 1993.

[Bar84] Henk Barendregt. *The Lambda Calculus; Its syntax and Semantics (revised edition).* North Holland, 1984.

[Dan90] Vincent Danos. *La logique linéaire appliquée à l'étude de divers processus de normalisation (et principalement du λ-calcul).* PhD thesis, Université de Paris VII, 1990. Thèse de doctorat de mathématiques.

[DCK96] Roberto Di Cosmo and Delia Kesner. Strong normalization of explicit substitutions via cut elimination in proof nets, 1996. Available from http://www.dicosmo.org/Publications.

[DJS93] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. The structure of exponentials: uncovering the dynamics of linear logic proofs. In Springer, editor, *3rd Kurt Gödel Colloquium*, number 713 in LNCS, pages 159–171, Brno, 1993.

[DJS95a] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. On the linear decoration of intuitionistic derivations. *Arch. for Math. Logic*, 33:387–412, 1995.

[DJS95b] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. On the linear decoration of intuitionistic derivations. In *Archive for Mathematical Logic*, volume 33, pages 387–412, 1995.

[DJS95c] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. Proof-nets and the hilbert space. In *Advances in Linear Logic*, number 222 in Leture Notes. London Mathematical Society, 1995.

[DR95] Vincent Danos and Laurent Regnier. Proof-nets and Hilbert space. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 307–328. Cambridge University Press, 1995. Proceedings of the Workshop on Linear Logic, Ithaca, New York, June 1993.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50, 1987.

[Gir90] Jean-Yves Girard. La logique linéaire. *Pour La Science*, (150):74–85, April 1990. French edition of Scientific American.

[Gir95] Jean-Yves Girard. Geometry of interaction III: The general case. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 329–389. Cambridge University Press, 1995. Proceedings of the Workshop on Linear Logic, Ithaca, New York, June 1993.

[GLT90] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Cambridge University Press, 1990.

[GM97] Stefano Guerrini and Andrea Masini. Parsing mell proof nets. 1997.

[GMM97] Stefano Guerrini, Simone Martini, and Andrea Masini. Parsing mell proof nets. In *TLCA*, 1997. Available as TR-96-34 dipartimento di informatica Pisa Italy.

[HS80] Roger Hindley and Jonathan P. Seldin. *Introduction to Combinators and λ-calculus*. London Mathematical Society, 1980.

[Jac94] Bart Jacobs. Semantics of weakening and contraction. *Annals of Pure and Applied Logic*, 69:73–106, 1994.

[Kan94a] Max I. Kanovich. Simulating linear logic with 1-linear logic. Preprint 94-02, Laboratoire de Mathématiques Discrètes, University of Marseille, 1994.

[Kan94b] M.I. Kanovich. The complexity of Horn fragments of linear logic. *Annals of Pure and Applied Logic*, 69:195–241, 1994.

[Laf95] Yves Lafont. From proof nets to interaction nets. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 225–247. Cambridge University Press, 1995. Proceedings of the Workshop on Linear Logic, Ithaca, New York, June 1993.

[Lin95] Patrick Lincoln. Deciding provability of linear logic formulas. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 109–122. Cambridge University Press, 1995. Proceedings of the Workshop on Linear Logic, Ithaca, New York, June 1993.

[LMSS92] Patrick Lincoln, John Mitchell, Andre Scedrov, and Natarajan Shankar. Decision problems for propositional linear logic. *Annals of Pure and Applied Logic*, 56:239–311, April 1992. Also in the Proceedings of the 31th Annual Symposium on Foundations of Computer Science, St Louis, Missouri, October 1990, IEEE Computer Society Press. Also available as Technical Report SRI-CSL-90-08 from SRI International, Computer Science Laboratory.

[LW94] Patrick Lincoln and Timothy Winkler. Constant-only multiplicative linear logic is NP-complete. *Theoretical Computer Science*, 135:155–169, 1994.

[Reg91] Laurent Regnier. *λ-calcul et réseaux*. PhD thesis, Université de Paris VII, 1991. Thèse de doctorat de mathématiques.

[Reg92] Laurent Regnier. *λ-Calcul et Réseaux*. PhD thesis, University of Paris VII, 1992.

[Rov92] Luca Roversi. A compiler from Curry-typed λ-terms to the linear λ-terms. In P. Mentrasti and M. Venturini Zilli, editors, *Proceedings of the Fourth Italian Conference Theoretical Computer Science*, pages 330–344, L'aquila, Italy, 1992. World Scientific Publishing Company.

[Sch91] Harold Schellinx. Some syntactical observations on linear logic. *Journal of Logic and Computation*, 1(4):537–559, 1991.

APPENDIX A

Proof of completeness for *ACC*

First of all, let's be precise about the notation we will use in this Section.

Definition A.1.1 (Sub-paired-graph) *Given a paired graph $S = (E, V, C)$, we say that $S' = (E', V', C')$ is a* sub-paired-graph *of S if*

- it is a subgraph of S (i.e. $E' \subseteq E$ and $V' \subseteq V$)
- C' is C restricted to E' (i.e. the only paired edges of S' are those edges that are already paired in S

Definition A.1.2 (Free pair) *A pair $c = \{a, b\}$ is called* free *if its basis is of degree 2 (i.e. if a, b are the only edges attached to the basis of c.*

Definition A.1.3 (Path, path composition) *A (finite) path α in a graph S is a (finite) sequence e_i of different edges of S s.t. e_n shares one node with e_{n+1}, different from the node shared with e_{n-1} in case $n > 1$. If β is a path that extends α, we note $\beta :: \alpha$ their composition.*

Definition A.1.4 (Pseudo-realizable path) *A path α in a paired graph S is* pseudo-realizable *(short,* p.r.*), if it contains only free pairs.*

Definition A.1.5 (Attractive Path) *A path α in a paired graph S is* attractive *if from every node u of S that does not belong to α there are at least two pseudo-realizable paths to α that are* distinct *in u (i.e. do not use the same edge when*

leaving u). We can restrict ourselves to paths that intersect α only at the end (i.e. they share just the last node with α).

Actually, in the proof of the theorem, we will use a more relaxed condition than *ACC*.

Definition A.1.6 (Weak *ACC*) *A paired graph S satisfies* Weak-ACC *if all the graphs in $GC(S)$ are acyclic.*

This more relaxed condition is preserved by subgraphs.

Lemma A.1.7 (Sub-paired-graphs) *If S satisfies* Weak-ACC*, then any sub-paired-graph S' of S satisfies* Weak-ACC.

Proof. Notice the only non-legal configurations in a correction-graph of a paired graph S is the presence of both edges of a pair. Now, if a path α is in a graph of $GC(S')$, then it cannot contain both edges of a pair c of S'. But now, by definition of sub-paired-graph, if S' contains both edges of a pair c of S, then c is a pair in S' too, so α cannot contain both edges of a pair c of S. This means that any cycle γ in a graph of $GC(S')$ would be a cycle also in some graph of $GC(S)$, hence the statement of the lemma follows. □

Theorem A.1.8 (Paired graphs with at least one pair are split (3.2.12)) *Let S be a* finite *paired graph that satisfies* Weak-ACC*. If S contains at least one pair then it is* split.

Proof. We will show that in any paired graph S s.t.

- (H1) all graphs in $GC(S)$ are acyclic,
- (H2) with a non-empty set of pairs $C(S)$,
- (H3) and with no splitting pair,

it is possible to find an infinite *strictly increasing* sequence S_n of sub-paired-graphs of S. By contradiction, the theorem will follow.
Actually, we will need to build also a sequence D_n of attractive pseudo-realizable cycles in each of the S_n. Let's remark now that in the proof we will always deal with *elementary cycles*, i.e. cyclic paths that do not contain already a cycle.
So, the inductive hypothesis we will use in the construction of the increasing sequence of sub-paired-graphs of S is the following:

- (Hi1) S_n is a sub-paired-graph of S,

- (Hi2) S_n has an attractive pseudo-realizable cycle D_n

Here comes the inductive construction.

- Base Case

  First, we notice that in any paired graph S satisfying H1, H2 and H3 we can actually find a cycle. Otherwise, any pair c is clearly a splitting pair (if there is no cycle in S, then it is actually a tree, and by removing the two edges of any pair c, we clearly disconnect the base from the other nodes of the pair). Take then an *elementary* cycle as the first sub-paired-graph S_0 of the sequence, and set $D_0 = S_0$.

  Notice that D_0 is trivially pseudo-realizable and attractive in S_0, since it coincides with S_0.

- Inductive Case

  Suppose now we are given S_n sub-paired-graph of S and D_n an elementary cycle that is pseudo-realizable and attractive in S_n.

  - Choosing S_{n+1}.

    First, notice that D_n must necessarily contain a pair c. Otherwise, there is a graph in $GC(S_n)$ that contains D_n, i.e. a cycle, that is impossible since S_n satisfies *Weak-ACC* by Lemma A.1.7. Furthermore, c is free in S_n because D_n is pseudo-realizable.

    Let's call x the base of c, and x_1, x_2 the other nodes of c. Now, c is not a splitting pair of S, by H3. This means there is a path in S connecting x with x_1 or x_2 not using the edges of c.

    Since c is *free* in S_n, such a path must necessarily use an edge e of S that does not belong to S_n. Let's call α the segment of this path that lies entirely in $S \setminus S_n$, but for x and the first node z of the path that belongs to S_n.

    Let S_{n+1} be S_n augmented with this path α, considered as a sub-paired-graph of S, so that Hi1 is trivially satisfied by S_{n+1}; notice that

    * S_{n+1} is strictly bigger than S_n, as at least the edge e was not in S_n
    * each new pair in S_{n+1} has at least an edge in α
    * the pairs in S_n that can happen to be no longer *free* have x or z as basis.

– Building D_{n+1}.

Now, we need to build D_{n+1}, pseudo-realizable and attractive in S_{n+1}. We will use α and D_n for this purpose. Figure A.1 will help in following the construction.

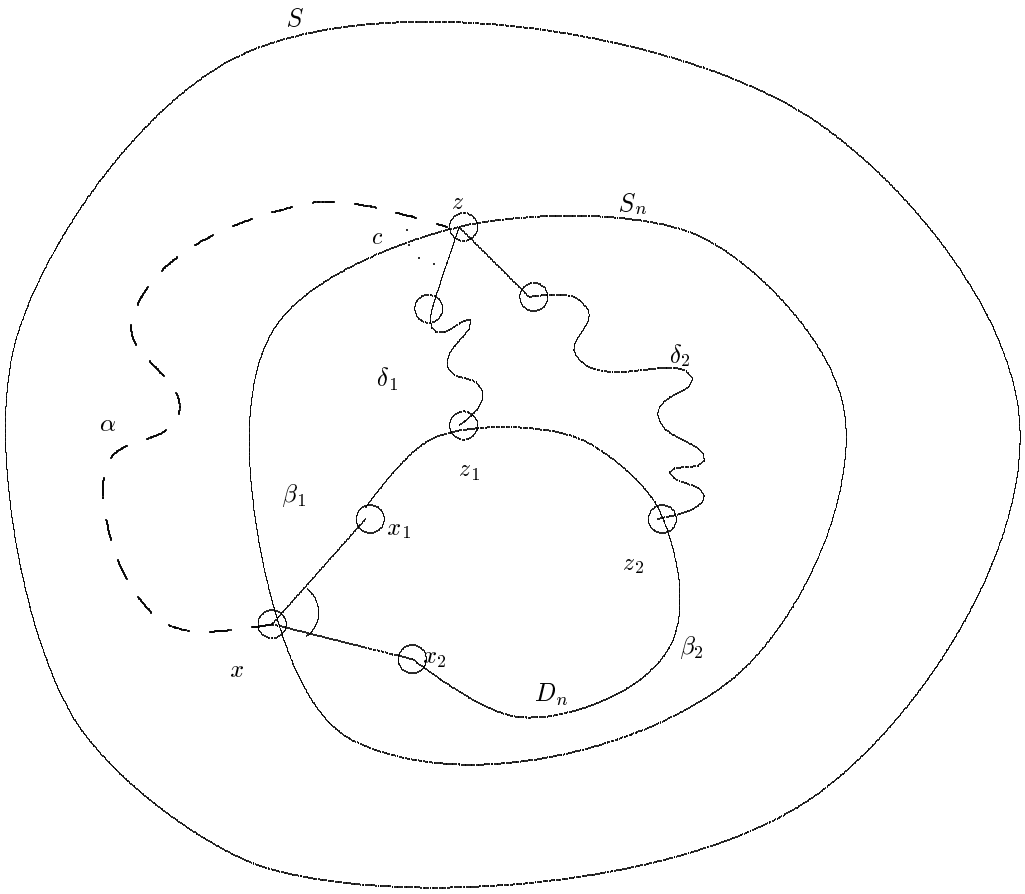


Figure A.1: The construction of D_{n+1}

There are two cases: either z belongs to D_n or not. We will assume first that z is not in D_n.

By Hi2, D_n is attractive, so there are two paths δ_1 and δ_2 from z to D_n, distinct in z and intersecting D_n only in one node.

Let's note these nodes z_1 and z_2, respectively. Now, z_1 (resp. z_2) cannot be x, as, by construction, in S_n, x has degree 2, while z_1 (resp. z_2) have degree 3. So, one of the following paths is pseudo-realizable:

$$\alpha :: \delta_1,\ \alpha :: \delta_2$$

Indeed, δ_1 and δ_2 are pr. in S_n, so they are pseudo-realizable in S_{n+1} as they use edges that were already in S_n, and the only possible pairs and edges we add in S_{n+1} can only modify the status of a pair based in z, which is not part of these paths. Also, α is p.r. by construction. The intersection of α and δ_1 (resp. δ_2) is z. Since δ_1 and δ_2 are distinct in z, $\alpha :: \delta_1$ and $\alpha :: \delta_2$ cannot both contain a new pair in z that was not in S_n.

So, one of the two junctions is p.r.; say it is $\alpha :: \delta_1$.

Call now β_1 and β_2 the two strict sub-paths of D_n connecting z_1 to x. One of the following cycles is p.r.:

$$\alpha :: \delta_1 :: \beta_1, \alpha :: \delta_1 :: \beta_2$$

The intersection of the paths $a :: \delta_1$ and β_1 (resp. β_2) is $\{z_1, x\}$. Furthermore, β_1 and β_2 are p.r. in S_{n+1} since they are p.r. in S_n and all their pairs are still free (remember that only the degree of z and x is affected by the construction).

Similarly as above, one shows that the junction in z_1 is correct (as β_1 and β_2 are distinct in z_1).

Since the only pair with base x belongs to D_n, the junction in x is correct in both cases. Suppose now that $\alpha :: \delta_1 :: \beta_1$ is p.r.: this will be our D_{n+1}.

In case $z \in D_n$, the same proceed applies, taking z in place of z_1.

– D_{n+1} is attractive. Now we need to check that D_{n+1} is attractive, i.e., for each u in S_{n+1} not in D_{n+1}, we need to show two pseudo-realizable paths distinct in u connecting it to D_{n+1} (again, we can consider that such paths intersect D_{n+1} only in the final node).

Since u is not in D_{n+1}, it is necessarily in S_n (as α, that contains the only new nodes w.r.t. S_n, is entirely in D_{n+1}).

If it is in D_n, since x and z_1 are in the intersection of the two cycles, the two strict sub-paths of D_n connecting u to x and z_1 are the required p.r. paths.

If it is not in D_n, there are two p.r. paths in S_n, ϵ_1 and ϵ_2, distinct in u, connecting it to D_n (by Hi2 for S_n).

Take ϵ_1. If it does not intersect D_{n+1}, it is easy to see that one can extend it in a pseudo-realizable path to x or z_1, giving the needed path. Otherwise, it intersects D_{n+1}, and the initial segment from u to D_{n+1} is the seeked path.

To check now that the two paths are distinct in u, it is enough to notice that in the construction we keep a non empty part of the ϵ, because u does not belong to D_{n+1}.

This ends the inductive part of the construction, so the theorem is proved.

This proof provides an effective procedure to find a pair contradicting H3, i.e. a splitting pair in S.
□

The Theorem3.2.12 follows trivially by the previous one, as *ACC* implies *Weak-ACC*.