Chapter 4

Computational aspects of the Embeddings into Linear Logic

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.

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 P is atomic $P^\star$ to be P.*

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 $M^A[x_1^{B_1} \ldots x_n^{B_n}]$ is turned into a proof-net $M^\star$ with conclusions ${B_1^\perp}^\star, \ldots, {B_n^\perp}^\star$ and $A^\star$.
Dropping the $\star$'s, here is the translation:

- 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:

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

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

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

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 $(\lambda x.x)y$ 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, but let us just say 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.

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:

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

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 ring 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:

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

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.* $⅋$*) to left premiss of a* $⅋$ *(resp.* $\otimes$*), and* does not *mix right with left.*

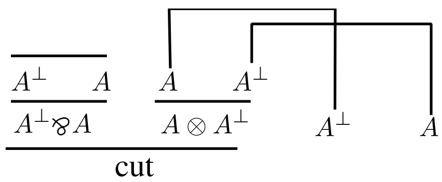
Computing the paths in P with the two matrices introduced above amounts to compose sections along cut-links, i.e., to build some path in P taking alternatively a section and a cut, which we weight with the product of the weights of the sections it uses. To put this in a formula:

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

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

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:



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, $F_1 = A^\perp ⅋ A, F_2 = A \otimes A^\perp, F_3 = A^\perp, F_4 = A$. 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 Vincent, est-ce que tu est d'accord ici?
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 M reduces to M', the nets associated to M and M' 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 the notes; 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

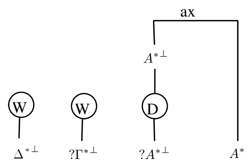
It is possible to extend this treatment of computation to the full typed λ-calculus, as we will briefly hint here, and event ot the pure untyped λ-calculus (see [Dan90, Reg92] for details). The first step is to define a translation suitable for non-linear terms, so we need exponentials, and we have many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

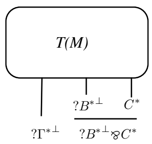
$$\begin{array}{lcll} A^* & = & A & \textit{if } A \textit{ is atomic} \\ (A\,B)^* & = & ?((A^*)^\perp)\mathbin{⅋}B^* & \end{array}$$

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

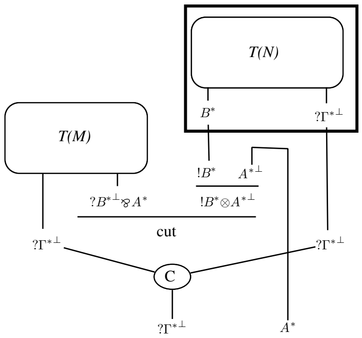
- *If* $\Gamma, x : A \vdash x : A$ *is an axiom, with* $\Gamma = C_1, \cdots, C_n$*, then its translation is:*



- *If* $\Gamma \vdash lx : B.M : B\,C$*, 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:

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

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

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

Remark 4.4.2 (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 4.4.3 (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 [AJ94b, Gir95]
% Master File: CorsoPisa.tex

4.5 Game semantics

A different way to look at the computational content of proofs has appeared recently, and is based on a correspondence that, paraphrasing the well-known Cyrry-Howard isomorphism, could be stated as the of *formulae as games, proofs as strategies* paradigm, and is known as *game semantics*. We will try here to outline the key ideas of this new paradigm, giving the appropriate pointers to the literature for who wants a detailed presentation, and focus on the results that show the connection between game semantics and the other approaches, like the dynamic algebra/geometry of interaction or even the abstract machines formalism.

Game semantics has recently produced two quite different-looking models: Hyland-Ong (henceafter HO) and Abramsky-Jagadeesan-Malacaria (henceafter AJM) and triggered new intuitions about syntax, while offering enough domain-theoretic structure to carry over the basic regulative tasks