

UNIVERSITÀ DEGLI STUDI ROMA TRE
DIPARTIMENTO DI MATEMATICA

TESI
per l'ottenimento del titolo di
Dottore di Ricerca in Matematica



UNIVERSITÉ PARIS DIDEROT (PARIS 7)
U.F.R. D'INFORMATIQUE

THÈSE
pour l'obtention du diplôme de
Docteur de l'Université Paris Diderot
spécialité informatique

UNIVERSITÀ
ITALIA
FRANCESE

Nets between Determinism and Nondeterminism

PAOLO TRANQUILLI

Direttori
Directeurs Lorenzo TORTORA DE FALCO / Antonio BUCCIARELLI

Data della discussione
Date de soutenance 23/04/2009

Commissione e revisori
Jury et rapporteurs Edoardo SERNESI, Presidente / Président
Antonio BUCCIARELLI
Thomas EHRHARD
Martin HYLAND
Simone MARTINI
Laurent REGNIER
Lorenzo TORTORA DE FALCO

Ringraziamenti

Remerciements

Acknowledgements

Voglio innanzitutto ringraziare Lorenzo Tortora de Falco per avermi appoggiato e guidato in questi anni. I suoi consigli e la sua amicizia sono stati un continuo impulso ad andare avanti.

Grazie anche ad Antonio Bucciarelli: ha accettato di codirigere questa tesi e mi ha aiutato molto in quel di Parigi. Oltre alla sua simpatia, ammiro molto (e, seriamente, non lo dico con ironia) il suo modo di scomparire quando parte.

Grazie a Marco Pedicini. Mi ha accompagnato nel primo anno del dottorato, e anche se poi ho preso una strada leggermente diversa, devo comunque molto a lui. Sul piano della ricerca spero ancora di poter lavorare con lui in futuro, se non altro per essere a distanza 3 da Erdős!

Thanks to Martin Hyland, Thomas Ehrhard and Simone Martini for taking the time in reviewing this thesis. To Martin goes also my gratitude for my stay in Cambridge early in my Ph.D, and his incredible willingness to discuss and give insightful pieces of advice. En outre de ma gratitude, mon admiration va à Thomas pour sa considérable activité de recherche, qui a donné la base pour la plupart de cette thèse.

Thanks to the other members of the jury, Edoardo Sernesi and Laurent Regnier, for accepting to take part in it. Cette thèse doit autant à la recherche passée et présente de Laurent, à lui vont donc ultérieurs remerciements.

Questa tesi è stata parzialmente finanziata dall'Università Italo-Francese nell'ambito del Programma Vinci, ringrazio dunque per il supporto ricevuto.

Ringrazio indistintamente tutti i “romani” della logica, oltre ai già citati Lorenzo e Marco: Michele Abrusci, Stefano Guerrini, Roberto Maieli, Beniamino Accattoli, Paolo Di Giamberardino, Damiano Mazza, Michele Pagani, Gabriele Pulcini. L'ambiente romano è stato sempre stimolante grazie a loro.

E dopo i ringraziamenti indistinti, ecco quelli distinti. Ringrazio Beniamino, tra l'altro, per come tira fuori dei discorsi interessanti anche senza nessuno spunto: averlo conosciuto in quel di Marsiglia è stato un ottimo valore aggiunto a questo dottorato. Ho sempre apprezzato la compagnia e la simpatia di Paolo, un'amicizia che, ti dico guarda, metti tipo lei contro un lavandino, guarda vince lei. Damiano, grazie per l'allegria, l'entusiasmo, l'occasionale aiuto, e come non nominare il Supciné Hebdo, di cui porto con orgoglio il titolo (vitalizio, non dimentichiamoci) di “frequentatore il più assiduo”. Ringrazio infine Michele sia

per gli interminabili discorsi di Scienza, faccia a faccia o su gmail, ma soprattutto per la sua amicizia. Non si direbbe che sia *così* vecchio :P.

Autres remerciements vont aux “français” rencontrés au PPS. L’ambiance de ce labo a toujours été plus qu’agréable, grace à tous eux.

Innanzitutto gli italo-foni, vuoi per nascita o per “infusione”, vi ringrazio anche se mi avete impedito in tutti i modi di imparare il francese (sì sì, è proprio tutta colpa vostra). Séverine Maingaud, grazie per il tocco di allegria dato all’ufficio. Giulio Manzonetto, un ottimo compagno, lo ringrazio con immutata stima. E tutti gli altri, mi raccomando, continuiamo l’invasione!

No i nawet mogę napisać coś po polsku, aby podziękować Juliuszowi Chroboczkowi: fajny kumpel, piwko z nim to zawsze przyjemność, jak również pracować dla niego jako chargé de TD.

Je remercie les autres qui ne parlent pas (assez) l’italien (pour le moment, je parie que ça ne durera pas longtemps) : toute la joyeuse troupe du bureau 6C12, particulièrement Stéphane Gimenez (merci pour les discussions “linéaires”), Christine Tasson (merci aussi pour le coupe des cheveux) et Gregoire Henry (merci pour l’instructif travail ensemble sur le TD); de l’autre côté du couloir Samuel Mimram (merci pour les incursions de contrôle sur mes activités à l’ordinateur); et enfin Philippe Hesse (merci, *man!*).

Altri ringraziamenti vanno a tutti gli amici di vecchia data, quelli che sento ancora e quelli che ho perso di vista. Non ne faccio la lista perché ne dimenticherei sicuramente qualcuno. La mia vita è fatta anche di voi e non vorrei mai altrimenti.

Un grazie speciale a mamma e a papà. Sulla mia tesi di laurea scrissi che se c’è qualcosa di buono in me, lo devo a loro, e qui lo ripeto. E grazie anche per la pazienza con la quale, quando ero in casa immerso totalmente nel lavoro, mi hanno sopportato e supportato in modo unico.

E un grazie enorme va a Mari Giò: questa tesi è spesso stata una rivale, ma il suo appoggio e la sua fiducia sono linfa per me. Le devo più di un semplice ringraziamento.



Contents

1	Introduction	1
1.1	Plan and Presentation of the Results	11
1.2	Notation	13
I	Tools, Questions and Aims	15
2	Nets	16
2.1	Statics	16
2.1.1	Ports, Cells and Wires	16
2.1.2	Sums	27
2.1.3	Boxes	30
2.2	Dynamics	33
2.2.1	Subject Reduction	35
2.2.2	Structural Congruence	35
2.2.3	Confluence	37
2.3	Invariants	38
2.3.1	Denotational Semantics by Experiments	39
2.3.2	Webbed Semantics and Semantic Correctness	41
3	Linear Proof Nets	43
3.1	The Multiplicatives	43
3.1.1	Sequent Calculus and Desequentialization	44
3.1.2	Correctness Criterion and Sequentialization	45
3.1.3	MLL nets as linkings	46
3.2	The Additives	47
3.2.1	Sequent Calculus and Desequentialization	48
3.2.2	MALL Nets as Sets of Linkings	49
3.2.3	Correctness Criterion and Sequentialization	51
3.2.4	Cut reduction	57
3.3	Invariants and Semantic Correctness	59
3.3.1	The Interpretation	59
3.3.2	Coherent Spaces	59
3.3.3	Hypercoherent spaces	64

4 Exponential Proof Nets and Lambda Calculus	69
4.1 Rewriting Theory Modulo Equivalence	70
4.2 The Exponentials	71
4.2.1 Correctness and Properties of Reduction	72
4.2.2 Accommodating the Contractions: Associativity, Neutrality, Push and Pull	74
4.2.3 λ -nets and λ -calculus	75
4.3 Differential Nets, Differential λ -nets and Resource Calculus	80
4.3.1 Differential Interaction Nets	80
4.3.2 Resource Calculus	82
4.3.3 Differential Nets and Boxes	84
4.3.4 Lax typing	88
II Determinism: Hypercoherent Spaces and Linearity	93
5 Hypercorrectness	95
5.1 The Criterion	95
5.1.1 Jumping from Contractions	95
5.1.2 $\&$ -oriented and Compatible Paths	96
5.2 Hypercorrectness Implies HCoh -correctness	98
5.3 HCoh -correctness Implies Hypercorrectness	102
5.4 Complements	104
5.4.1 Cut Exposure	105
5.4.2 Stability Under Reduction	106
III Nondeterminism: Differential Operators and Resources	109
6 Church-Rosser Theorem for Pure DiLL	110
6.1 Marking New Cuts: DiLL°	110
6.2 Finiteness of Developments	112
6.2.1 Measuring Exponential Reduction	112
6.2.2 The Proof, Case by Case	120
6.3 Confluence of Developments	126
6.3.1 Local Confluence Modulo	127
6.3.2 Local Coherence	130
7 Standardization and Conservation Theorems	133
7.1 Proof of the Standardization Theorem	135
7.2 $\text{DiLL}_{\partial\varrho}$	136
7.2.1 Labelled Nets and Reduction	137
7.2.2 Confluence of Labelled Non Erasing Reduction	139
7.3 Proof of the Theorem	140
7.3.1 Conservation for $\text{DiLL}_{\partial\varrho}$	142
7.3.2 Transferring Conservation from $\text{DiLL}_{\partial\varrho}$ to DiLL	146

8 Full Resource Calculus	154
8.1 The Language	154
8.2 Giant Step and Baby Step Reductions	155
8.3 The Translation	160
8.3.1 Statics: Definition and Sequentialization	160
8.3.2 Dynamics: Bisimulation	162
Index	170
List of Figures	172
Bibliography	174
Reference Figures	179

Chapter 1

Introduction

A team of sociologists decides to conduct an experiment. Inside an empty room they put a gas stove and a table. On the table they put a lighter and a saucepan with some water in it. Then in succession they let into the room a physicist, an engineer and a mathematician, with the task of boiling the water.

The physicist begins to fiddle with the stove, turning the knobs. He hears and scents the hissing of the gas. He turns his attention to the lighter, turns it over in his hands, and after some trials he lights it. He returns to the stove, approaches the lighter to the open gas, and slightly burns himself in a small burst of flames. Sucking on his finger, he puts the saucepan on the stove and boils the water.

The engineer looks around, opens a drawer of the stove and searching through it he happily comes out with an instructions handbook. He reads through it, paying particular attention to security warnings. After picking up the lighter, he lights it near the cooker and then opens the gas. He puts the saucepan upon the flame and boils the water.

The mathematician takes a look and then begins to think for a long time, at times walking nervously around the room, at times scribbling on a crumpled piece of paper he accidentally found in his pockets. Then, alight with understanding, he purposefully gets the lighter, lights the stove and boils the water.

After some time, the sociologists conduct the experiment again, this time putting the saucepan and the lighter on the floor.

The physicist gets the lighter from the floor, turns the flame on without incidents, lifts the saucepan on the stove and he is done.

The engineer remembers the instructions he read and does the same.

The mathematician, as soon as he grasps the situation, without further ado picks the lighter and the saucepan, puts them on the table in order to reduce to a problem he has already solved, and then boils the water, quod erat demonstrandum. \square

Before explaining why we started with this light joke, let us introduce the main question around which this thesis revolves.

What is the meaning of computational correctness?

Computation, which in its most intuitive sense is using a set of instruction in order to accomplish something, is a concept as old as mathematics, if not older. However one must wait until the beginning of the previous century before the subject gets an organic approach, whence comes the field of computer science.

It must be stressed that this came quite before actual modern computers came into being, and that the primary concern at the time was of logical nature: Hilbert's program for a clean foundation of mathematics. And it should also be noted that such early theoretical endeavours were not independent of the later development of digital computers. It is not a coincidence that the mathematician usually designated as the father of computer science, Alan Turing, deeply involved in Hilbert's program, was also involved in the design and project of the first digital computers.

From a certain point of view, correctness is what computation is all about: a procedure is correct if blindly following it we get an expected answer or behaviour. So the question turns into another form: what do we expect of a program? Various answers are possible, depending on what are the practical or theoretical aims.

Modus ponens and cut elimination. Before going into what are the particular issues and answers contained in this thesis, let us introduce our background, returning to our mathematician concerned with boiling water.

What did the mathematician just do? He blindly applied the *modus ponens* rule of reasoning, formally described by

$$\frac{A \Rightarrow B \quad B \Rightarrow C}{A \Rightarrow C}$$

If we have that B ("lighter and saucepan on the table") implies C ("boiled water"), and moreover that A ("lighter and saucepan on the floor") implies B , then we can combine the two "proofs" (for $B \Rightarrow C$ taking the lighter and the saucepan from the table, using the former to light the stove and putting the latter on, for $A \Rightarrow B$ lifting lighter and saucepan from the floor to the table) and obtain one for $A \Rightarrow C$. In proof theory such rule is called *cut* since Gentzen's sequent calculus [Gen67], as it cuts away a proposition (B) from the result of the reasoning. In mathematics this is the basic concept of applying a lemma to get a new result.

What did the physicist and the engineer do? They too (at least unconsciously) applied modus ponens by relying on previous experience. However they did not provide the actual "formal" composition of the two proofs, but rather a *reduced* one, going directly from A to C . Gentzen's most important result is his *Hauptsatz (fundamental theorem)*: if A is provable in intuitionistic or classical logic, then it is so without the use of cuts. In particular the absurd or contradiction \perp , which can only be introduced by a cut, cannot be proved, and the logical system is *consistent*. Gentzen's objective was in fact proving the consistency of arithmetics.

One heritage of Gentzen's theorem is its proof. He provided a *reduction* procedure that, given a proof π of A with cuts, gives another proof π' of A , such that we can iterate this step and eventually *normalize* to a proof without cuts. Such procedure is the *cut elimination*, and the main contents of the proof of the Hauptsatz is that we can make cut elimination terminate, arriving to a normal form, i.e. a proof from which we cannot proceed further, which is necessarily cut free.

So, modus ponens and lemmas are useless? Quite the contrary. We may say that lemmas are the true creative and intelligent contents of rigorous and non rigorous thinking. Without going into the philosophical and psychological implications, the evident advantages of reasoning by lemmas are the possibility of taking a lemma “as is”, like a black box, without having to reprove it, and most importantly of using it as many times as we please. This reuse of lemmas is granted by the *structural rules* of contraction and weakening. We will come back to those later.

From the point of view of a human prover and actual mathematical proving, cut elimination turns in fact an intelligent proof into a dumb, repetitive and usually huge one, where every single statement is reproved every time it is used, and all are traced back to the founding axioms of the theory.

So, is cut elimination useless? Not at all. Let us return to our mathematician. After he has obtained his proof of $A \Rightarrow C$, he has A , the actual situation with saucepan and lighter on the floor. The two can be combined with another cut rule to obtain a proof of C . But he is not content with just having this proof, something like “it is possible to boil the water”. He *executes* it, following his proof (so passing by the table) and arriving to what we can see as the cut free proof of “boiled water”: that specific water, actually boiled.

Curry-Howard

Proofs as programs. This suggests that the mathematician's proof on the scribbled piece of paper can be in fact regarded as a program, and cut elimination is the execution of such a program. In fact the consequences of the Hauptsatz originated another historic leap forward in proof theory, one that reconciled logic and computer science, long after the beginning of the century had seen them so close. We are speaking about the *Curry-Howard* isomorphism [How80], after observations of Curry in the '30s and '50s, and Howard in the late '60s.

Their core observation was that there was a logical system, a fragment of natural deduction, which was in complete correspondence with Church's typed λ -calculus [Kri93], the functional model of computation at the time seemingly unrelated. The achievements are the following.

- *Proofs as programs*: a bijection between proofs of natural deduction and λ -calculus programs.
- *Propositions as types*: a proof of a formula A is mapped to a λ -term of type A . In particular the proof of an implication $A \Rightarrow B$ correspond to a program of type $A \rightarrow B$, i.e. one taking objects of type A as inputs and outputting objects of type B .
- *Cut elimination as execution*: a step of cut elimination of a proof in natural deduction corresponds to a step of β -reduction of the related λ -term.

The Curry-Howard isomorphism gave rise to a common ground between mathematical logic and computer science that has sprouted and is continuing to sprout a wide spectrum of research, fruitfully bridging results and techniques from one side to the other.

Correctness by specification. After this long detour we come again in touch with our central topic. One of the early research direction in this field was *program extraction*. A notable result in the wake of the Curry-Howard isomorphism is that given a function f defined in logical terms then from a proof of totality, i.e.

$$\forall x \exists y : N x \implies N f(x),$$

where $N x$ is the predicate stating that x is a natural number, one can extract a program calculating the function f [KP90] and always terminating. In fact, the program *is* the function. The logical description of f a *specification* of a program, i.e. the conditions that the output must satisfy given a certain input. In other words, a condition of correctness, in the sense that it does what it is expected to do. Here the Curry-Howard correspondence ships a proof of correctness as the program calculating f . This is about the correctness of a single program, but what can we say of a general logical system or programming language?

Correctness by good reduction behaviour. If we shift our viewpoint to *rewriting theory*, then we see both cut elimination and program execution as a rewriting relation on some language. We thus naturally inherit a number of other notions that can be taken singularly or altogether as a measure of the correctness of a logical system or a programming language:

- the already mentioned weak normalization, i.e. every proof/program can be reduced to a normal form, a cut free proof/value;
- *strong* normalization, i.e. no matter how we reduce a proof/program, the cut elimination/computation always terminates;
- confluence, also called Church-Rosser property, i.e. if we reduce in two different ways we can reduce to a common form; in particular if there is a normal form then it is unique, i.e. normalization is *deterministic*.

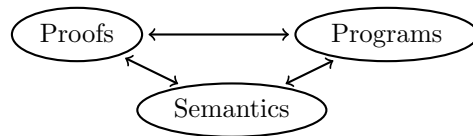
As a side note, the last point was proved for λ -calculus for a logical reason, in order to prove consistency of the theory of λ -calculus, which was a try at, again, giving solid mathematical foundations along Hilbert's program.

In computer science *type theory* is the field that studies just what correctness can be imposed on programs by typing them in some logical system (in fact, establishing instances more or less strong of the Curry-Howard isomorphism). This is a standard tool to certify that a program has correctness properties like the three we mentioned. Examples of the achievements on this line are the functional programming languages, such as CAML.

Invariants. As for both proofs and programs studying the reduction becomes the most important issue, a great importance is given to the study of *invariants* of the transformation given by reduction. For example if a system has weak

normalization and confluence, then the normal form is an invariant. However, in order to find tools that truly add something in the picture, one really needs to look for some solid foundation outside the system.

In logic and computer science the study of invariants of reduction is called *denotational semantics*, with Scott domains [Sco72] among its first achievements. Its basic idea is to interpret formulas/types A with some kind of mathematical structure $\llbracket A \rrbracket$, and a proof/program of type $A \rightarrow B$ with a morphism between the two structures $\llbracket A \rrbracket \rightarrow \llbracket B \rrbracket$. The requirement will then be that if π reduces to π' , then $\llbracket \pi \rrbracket = \llbracket \pi' \rrbracket$. Denotational semantics can therefore be inserted as a third agent of the Curry-Howard correspondence, giving a (possibly) convincing mathematical framework to the dynamics of the systems.



Again the three sides can act as a bridge between all the actors involved, transporting notions, results and also questions between a domain and the other, as we hope will be made clear by some examples we will show next.

Linear Logic

From invariants to linear logic. While the three-sided Curry-Howard correspondence described above flourished for natural deduction, intuitionistic logic and functional programming, sequent calculus remained for long out of the picture. Its main shortcomings are the absence of the strong normalization property and more importantly confluence, so that convincing invariants for reduction are hard to find. In fact for Gentzen's sequent calculus the only invariant is provability: i.e. all proofs of A are identified. Such a behaviour can be traced back to an inherent non determinism of cut-elimination: reducing a proof of A may give different normal forms by non confluence.

But it was right from the semantical analysis of intuitionistic logic that came an answer: the discovery by Girard of linear logic (LL, [Gir87]). The observation was based on the fact that on *coherent spaces* and stable functions, a model of system F, second order intuitionistic logic, the implication can be decomposed into two different operations

$$A \rightarrow B = !A \multimap B$$

where

- $A \multimap B$ are semantically *linear* morphism from A to B ;
- $!A$ denotes in a sense a space of reusable "packages" of A .

On the semantics side the morphisms $!A \multimap B$ should be entire analytical functions, though at this stage the correspondence is shaky, and we will come back to it later.

Why linear? This behaviour of the semantics side of the correspondence, where linearity in the mathematical sense is (somewhat) recovered, can be lifted to the other two sides. Summarizing: linearity corresponds in logic to forbidding the structural rules weakening and contraction, and in computer science to using inputs exactly once.

Linear logic is the logical system that acknowledges such distinction, relegating the structural rules only to formulas explicitly introduced to accept them, the exponentials. The system then is seen to split the connectives in two classes, the *multiplicatives* \otimes and \wp (*tensor* and *par*), and the *additives* $\&$ and \oplus (*with* and *sum*), with \otimes and $\&$ playing the role of conjunctions and \wp and \oplus that of disjunctions. Then we have the two *exponential* modalities $!A$ and $?A$ (*bang* or *of course* and *why not*), with structural rules allowed on $?A$. However the most important operation is arguably the involutive negation A^\perp (the *dual*) for which the above connectives satisfy De Morgan’s duality, each in their class.

Duality and Proof Nets. It is evident in classical logic that a proof of $A \Rightarrow B$ proves also $\neg B \Rightarrow \neg A$. However if such proof effectively turns hypotheses A into the thesis B , in general we will not be able to see the same in the inverse, i.e. an effective way of turning $\neg B$ into $\neg A$. The structure of a proof of $A \Rightarrow B$ can be seen as a tree with multiple leaves for where the A hypotheses are used in the proof. Turning such tree upside down is far from giving the same picture.

On the other hand a proof of $A \multimap B$ is one that uses exactly one hypothesis A to provide exactly one thesis B . We can intuitively see the significance of having a duality with respect to such arrow: $B^\perp \multimap A^\perp$ is again the space of proofs bringing one B^\perp into one A^\perp . This leads to see that there is no preferred order in such proofs: proof *trees* can become a more general object with a more relaxed structure, proof *nets*.

In this new syntax proofs of sequent calculus get quotiented with respect to commutation of logical rules in the tree. As such it is a desequentialized version of sequent calculus, much closer to the computational contents of the proof.

As the exponentials break the pure linear nature of the logic, in proof nets they are responsible for the appearance of *boxes*, i.e. areas of the proof that are in a way marked to be together and not dispersed by the unsequential nature of the nets.

Back to computer science. As we have already said, linearity in computer science corresponds to arguments which are used only once, i.e. that are not duplicated or erased. Linear logic gives a precise logical framework where one can dissect the behaviour of programs on this issue. For an example of the novelty of this approach we have logical systems for implicit computational complexity, the so called light logics (the first being light linear logic [Gir98]). To put it in Girard’s words “the abuse of structural rules may have damaging complexity effects”. This gives by the way yet another possible interpretation of correctness: the one relative to good complexity behaviour, as poly or elementary time, which is granted by typing in light logics.

Another example, more strictly related to proof nets, is the fact that as intuitionistic logic translates into linear one, we recover a translation of λ -terms in proof nets, where we can profit from the local and desequentialized nature of such system. Danos [Dan90] and Regnier [Reg92] studied this translation,

arriving to define an untyped version of proof nets that give to λ -calculus a syntax akin to Lamping graphs for optimal reduction, though in a completely logical setting.

In such translation exponential boxes become the arguments, so that they can be used as many times as is necessary. We get here the intuition that a box is a sort of package, marked so that we know what must be copied or deleted when such operation must be executed.

Discerning correctness among mistakes. One great novelty of proof nets is that they live in a richer syntax, **nets** (usually called proof structures in the literature [Gir87]), where cut elimination is defined. Basically, it is a perfectly defined syntax where errors are allowed, beyond just typing mismatches. This opens a totally new point of view in regarding correctness. After all, giving a uniform setting in which we can do mistakes can indeed add up to the study of what is not a mistake.

In proof nets one typically gives a geometric criterion characterizing “true” proof nets. What can be asked from such criterion?

- *logical* correctness: the net is indeed a proof, which means that it is the desequentialization of a sequent calculus proof;
- *dynamical* correctness: the criterion is invariant (*stable*) under reduction, which means we can truly compute with these objects;
- *semantical* correctness: the criterion tightly corresponds to the semantics.

While the first two points are woven in the twenty years of linear logic tradition, the third is quite a recent point of view. It makes sense due to the fact that not only cut elimination but also semantics is perfectly defined on nets, giving in general plain sets. Then the semantics of choice can distinguish “right” objects out of those sets. For example coherent spaces, which are just unoriented graphs built upon sets called *webs*, distinguish a set of atoms of the web if it is a *clique*, i.e. points pairwise coherent. Such investigation is thus in the spirit of ferrying properties and notion in the three-sided correspondence between logic, computer science and mathematics. For MLL there is the early work by Retoré [Ret97], while our most recent point of reference is Pagani’s one for MELL [Pag06a] and *differential linear logic* [Pag08], a system we will introduce up next.

During the preliminary Part I we will address the question of which relation exists between these different notions, taking the multiplicative additive fragment of LL and Ehrhard’s hypercoherent spaces [Ehr95] as the targets of our investigations, whose results will be the object of Part II.

Differential Linear Logic

Back to invariants: vector spaces. Let us go back to linearity and analyticity in semantics. Such notions clearly need a linear algebra setting, a setting from which linear logic has always borrowed its jargon. Ehrhard has introduced in [Ehr02, Ehr05] new models of linear logic that convincingly use such setting.

Formulas get interpreted by vector spaces, while proofs (and programs) by continuous linear maps. Such intuitions were indeed already present in coherent spaces. Namely, the operations on webs underlying tensor, dual or (direct)

sum of coherent spaces are the same done on bases in their counterparts of finite-dimensional vector spaces. However the exponential modality breaks this parallel as it necessarily introduces infinite webs, and thus infinite bases.

It is well known in linear algebra that such infinite dimensional spaces have a series of quirks with respect to their finite dimensional cousins, unless we endow them with some kind of topology. The works of Ehrhard succeed in finding one good enough for the purpose of modelling linear logic. In the spaces of [Ehr02, Ehr05] the correspondence between the connectives and the operations on topological vector spaces becomes totally precise, so that

- the tensor \otimes is indeed the tensor product of two spaces;
- the linear implication \multimap gives the space of continuous linear maps;
- the dual A^\perp gives the dual in the topological sense, i.e. the space of linear continuous maps from A to the field (which therefore takes on the role of the unit \perp).
- the finite cartesian product $\&$ and coproduct \oplus get identified in the direct sum of vector spaces; again, just like in vector spaces, the two get distinguished in the infinite case in the infinite direct product and sum of spaces respectively.

Finally, morphisms in $!A \multimap B$ can be seen as analytical functions. As such, we may glint yet another refinement of the intuitionistic arrow. An analytical function can in fact be seen as its Taylor expansion, i.e. as an infinite sum of multilinear forms. This expansion is obtained by iterating derivation on such morphism, a notion that makes perfect sense in this semantical setting.

Can all this be lifted by the three sided correspondence (semantics, logic and computer science) to the other two fields? The affirmative answer came from Ehrhard and Regnier who presented extensions with syntactic differential operators for both linear logic [ER06b] and λ -calculus [ER03].

Back to computer science: derivating programs. What does taking a derivative mean? One explanation is that it gives the best linear approximation of a function in a point. Intuitively we may think of a way of getting such a linear approximation of a function *without* looking into it: feeding it with an input that is inherently linear, and just see how the function handles it.

This is exactly the behaviour in programs that the derivative corresponds to by the three sided Curry-Howard. Given a program t of type $!A \multimap B$ derivating it means feeding it a linear argument u which can be used exactly once. Let us stress the difference with the linear functions $A \multimap B$ we already wrote about. Those are functions where the input is structurally bound to be used once. Here instead we are speaking about an ordinary program that uses multiple times its input, which finds itself with a one-use argument.

In this sense $\frac{\partial t}{\partial x} \cdot u$ is indeed the linear approximation of t applied to u , as u *must* be used in the usual computer science sense of linearity. This naturally gives non-determinism, as there is a choice as to where such one-use u should go among the possibly many queries for x . Such non-determinism exactly corresponds to the sum of vector spaces, therefore it is natural to model it on the syntactic level with the use of formal sums.

Let us see a simple parallel with calculus to explain the above point: if we consider the function x^2 , we can see it as the bilinear form $x_1 \cdot x_2$ applied to x and x again, so that we can see the distinction between the two linear occurrences of x . Then taking the derivative along a can be seen as non deterministically choosing one of the two occurrences and substituting a for it, i.e. $a \cdot x_2 + x_1 \cdot a$. Contracting back together the occurrences, we get the known $2x \cdot a$.

So it should be clear by now that while linearity was bridged to the concept of unduplicable and unerasable input, derivation does so to the concept of depletable input. There already were a few efforts to study such computational behaviour (like Boudol's calculus with resources [Bou93], on which we will return in Part I and then in Part III, or Kfoury's linearization of λ -calculus [Kfo00]), however we think that such study may take new life from the link with these novelties in linear logic, much like the study of linearity in λ -calculus did after Girard's discovery.

Taylor expansion. The correspondence of derivation has a deep consequence. Iterated derivation may be used to extract the Taylor expansion of an analytical function, and the same can be done for a term of λ -calculus. We can perform such expansion t^* inductively by setting $x^* = x$, $(\lambda x.s)^* := \lambda x.s^*$ and finally

$$((u)v)^* = \sum_{n=0}^{+\infty} \frac{1}{n!} \langle u^* \rangle (v^*)^n \quad (1.1)$$

where $\langle u \rangle v$ is the linear application of resource calculus, a fragment of differential λ -calculus dealing *only* with one-use resources, strongly related to Boudol's one, hence the name. Then all terms of ordinary λ -calculus are indeed analytical in a precise sense [ER08, ER06a]. Application is an operation which is linear in the function but not in the argument: the Taylor expansion may be seen as decomposing this non linear operation lying behind $!A \multimap B$ into an infinite sum of linear operations.

Back to logic: derivating proofs. Just like ordinary λ -calculus can be endowed with differential operators, so are proof nets. Just like the non linear application of λ -calculus can be decomposed in an infinite sum of linear pieces, so can the non linear exponential box of LL. This requires to give linear rules for introducing and managing the exponential modality $!$. Surprisingly, the result is a system where $!$ and $?$ are completely symmetric, just like the linear \otimes and \wp are. So as $?$ was introduced from a linear formula with the *dereliction* rule, $!$ was introduced out of nothing with *weakening* one and joined with *contraction*, we have the symmetric rules for $!$: *codereliction* (which represents derivation in 0), *coweakening* and *cocontraction*.

In these nets, introduced as differential interaction nets in [ER06b], we find, twenty years after LL was discovered, that indeed the exponential box is... the exponential. In fact provided certain conditions one can Taylor expand the nets

of LL, as shown in [dC07]. We then have

$$\begin{array}{c} \boxed{\pi}^* \\ \downarrow \end{array} = \sum_{n=0}^{+\infty} \frac{1}{n!} \overbrace{\begin{array}{c} \pi^* \dots \pi^* \\ \downarrow \quad \downarrow \\ \downarrow \quad \downarrow \end{array}}^{n \text{ times}}$$

The symbols will be explained later in the thesis. If however we take the addends to be powers of π^* (in a sense related to a convolution product), we exactly get the known expansion of the exponential. In this light the exponential isomorphism, known since the beginning of linear logic, stating the equivalence $!A \otimes !B \cong !(A \& B)$, is explained by $e^x \cdot e^y = e^{x+y}$. By the way, we will deal with a syntactic counterpart to such isomorphism in Part I and II.

Differential interaction nets and π -calculus. Though devoid of what made LL so expressive, the exponential box, this system has proven to be a highly interesting one in itself. In [EL07] Ehrhard and Laurent have translated into it a relevant fragment of calculus of processes, the finitary π -calculus. The dual rules of dereliction and codereliction take then the meaning of sending or expecting a signal on a channel, with contraction and cocontraction acting for routers, and weakening and coweakening turning off the channels.

We have seen how such logical framework seems to be apt at describing non determinism by depletable resources. This other result shows promise in also capturing another kind of non determinism, the one due to concurrency. It is the first time that we get a glimpse of the Curry-Howard correspondence in the realm of non-deterministic computation, and it could be the starting point for a fruitful analysis, if not renewing, of the paradigm.

Mixing linear and exponential. Though the seminal paper of differential interaction nets deals only with the linear exponential rules, the definition of the extension to the ordinary box of linear logic is straightforward, giving rise to *differential linear logic*. Even though such box can be replaced by its Taylor expansion, such procedure totally voids the system of its finitary (or, we would say, *uniform*) nature.

So, even though there is not yet a publication similar to Girard's [Gir87] for differential linear logic, there is already a lot of ongoing research being done on this system. Some first results may be seen in Vaux' extension to the polarized paradigm in [Vau08], or the already cited work on differential linear logic and finiteness spaces by Pagani [Pag08].

Reduction correctness in differential linear logic. However, due to the youth of this research area, there is still a lack of fundamental results about the good computational behaviour of the system, its correctness from this point of view.

Clearly the first question that comes to mind is whether the system is normalizing, clearly setting oneself in the typed case. Pagani in [Pag09] shows weak normalization for first order differential linear logic. Can we infer more?

What can be said when we abandon types? Such a setting is ideal for showing results which persist whatever the type system one might later put on the objects, though it is clearly a harder one. Many results of rewriting theory rely on termination. Most importantly, confluence becomes a subtler issue.

We have strived to fill this gap, and our results are the object of Part III. These achievements may be seen as parallel to the ones given by Danos and Regnier for MELL in their theses [Dan90, Reg92] in the wake of LL and proof nets, and by Pagani and Tortora de Falco much later in [PTdF08] for the whole of LL.

As a sidenote, one could wonder why one searches confluence in a non deterministic paradigm. The catch is that confluence of sums ensures that the divergence of reduction is due to the inherent forking of single steps, rather than by a choice of which part of the program to reduce.

To illustrate how such reduction correctness can be employed, we studied the natural extension of resource calculus (as presented by Ehrhard and Regnier in [ER06b]) to a mixture of depletable and perpetual arguments. We will see in Part III how we can translate this calculus into differential proof nets, and how the results proved about their reduction correctness can be lifted to such calculus.

1.1 Plan and Presentation of the Results

In the first part of the thesis we will pose the formal framework in which all our successive endeavours will take place.

In [Chapter 2](#), we will give the unifying definition of nets. We will thus introduce *nets*, *slice nets* (sets of nets, like MALL nets) and *polynets* (multisets of nets, like differential interaction nets). Great care is given to the formal definition also of exponential *boxes*, containing other nets or polynets. The definitions revolve around the concept of *context*, which permits to easily define reduction of nets, which is done in [Section 2.2](#). We also provide a very general formal definition of set denotational semantics for nets, and webbed semantics with, most importantly, the first definition of semantic correctness ([Section 2.3](#)).

In [Chapter 3](#), we present the systems of MLL and MALL inside the framework built in the previous one. The logical correctness of the system is stressed, by presenting the *correctness criteria*, the Danos-Regnier one for MLL, and its extension by Hughes and van Glabbeek for MALL, together with their sequentialization theorems. In the last part of the chapter we speak about the invariants of these system, introducing *coherent* and *hypercoherent* spaces. The relation between semantic correctness and MLL correctness is stated ([Theorems 3.16](#) and [3.18](#)). We also give a polished proof of the first which is the starting point for the proof of [Theorem 5.7](#) in [Chapter 5](#). The positive and negative results that propelled our research are presented.

In [Chapter 4](#) we switch to MELL and DiLL. Our viewpoint changes: we define logical correctness, but our interest does not lie in the logical contents like the sequentializability theorem, but rather in the good reduction properties it entails. We thus begin with a short introduction to more advanced rewriting theory notions, namely normalization and confluence modulo an equivalence relation. We then introduce MELL, its correctness criterion, and the intuitionistic pure nets, which we call λ -net to stress their tight relation with λ -calculus. Such

relation, as studied by Danos and Regnier, is presented in Section 4.2.3.

The [second part](#) of the chapters deals with the status quo for differential proof nets. We thus present: Ehrhard’s and Regnier differential interaction nets (DiLL_0 , as they are the 0-depth fragment of DiLL), and their relation with resource calculus, a linear revisiting of Boudol’s calculus. We finally move on to defining the syntax of differential proof nets with boxes. We will define a set of (optional except the associative one) equivalence relations, among which the exponential isomorphism equating a box with a um inside with the “multiplication” of two boxes. Motivated by the net shown in Figure 4.12, a counterexample to the striction lemma (stating that a pure net is SN iff it is WN by non erasing steps, a property fundamental for normalization proofs) we conclude the section by defining a form of typing we call *lax*, which does not exclude any net but rather excludes reductions, namely on those cuts that somewhere before were clashes.

In the second part of the thesis, [Chapter 5](#), we present a new criterion on MALL nets, *hypercorrectness* (Definition 5.1). This is a geometric characterization of semantic correctness for hypercoherent spaces, and comes also in an apparently weaker form which we call weak hypercorrectness. The latter is seen to be directly implied by Hughes and van Glabbeek’s correctness (Proposition 5.2).

The following section is devoted to the proof that weak hypercorrectness implies HCoh -correctness for saturated MALL nets (Theorem 5.3). Combined with Proposition 5.2 this gives the first proof of semantic soundness of MALL proof nets entirely based on paths in the net. Next, we prove that HCoh -correctness of a cut free MALL net implies its strong hypercorrectness (Theorem 5.7). In the last section we establish by semantical means that the two form of hypercorrectness are equivalent also in the case of nets with cuts (Proposition 5.10) and more importantly that hypercorrectness is stable under reduction (Theorem 5.11).

The third part of the thesis begins with [Chapter 6](#), completely devoted to prove the Church-Rosser theorem for pure DiLL proof nets (modulo equivalence). We arrive at such result by defining a parallel reduction which is strongly Church-Rosser (confluency and Church-Rosser are not equivalent in reductions modulo, see Section 4.1). In order to prove this, we need a finite development theorem (Theorem 6.1, which takes the most part of the chapter. Such theorem is rephrased as SN and $\text{CR}\sim$ for a system, DiLL° , that locks “new” cuts. The first is obtained by means of a decreasing measure (Proposition 6.8), the second by using a variant of Newman’s Lemma due to Huet, thus checking local confluence and local coherence with \sim (Proposition 6.9).

[Chapter 7](#) goes on and contains the proofs of the standardization theorem and the conservation one, in λ -calculus’ jargon. The first states that every reduction chain can be turned into a standard one, where no reduction is performed if it involves something that was already reduced. The second states the equivalence of strong normalization and the weak one for non erasing reductions, for any given pure DiLL proof net. The proof is carried out using Gandy’s method: if a term is normalizable and confluent, and we define a strictly increasing measure, then the term is strongly normalizing. However in order to do this one needs confluence of non erasing reduction, at which DiLL fails. We therefore pass via another system, $\text{DiLL}_{\partial\varrho}$, which replaces symmetrically codereliction and dereliction with new cells ∂ (*linear substitution*) and ϱ (*linear query*), using translations to bring back and forth results between the two systems.

Finally in [Chapter 8](#) we present the *full resource calculus*. This is a calculus having exactly the syntax of Boudol's λ -calculus with resources, with mixed perpetual and depletable resources. The difference is the reduction, an algebraic one directly taken from differential λ -calculus. A notable difference is that we are able to totally rule out sums inside arguments, by converting on the fly a perpetual sum in various perpetual resources, by setting $(^\infty u + v) = u^\infty v^\infty$.

As with resource calculus, the target of the Taylor expansion of ordinary λ -terms, reduction comes in two flavours, *giant step* $\xrightarrow{\mathbf{g}}$ and *baby step* $\xrightarrow{\mathbf{b}}$. However, having fully recovered the whole of λ -calculus, results on such reduction again become non trivial. However, a translation is established between such terms and DiLL λ -nets, t° . Then it is shown that such a translation is bijective ([Theorem 8.8](#), on *ec* normal nets without exponential axioms) and that two strong forms of simulation hold. First, $s \xrightarrow{\mathbf{g}} t$ if and only if $s^\circ \xrightarrow{\mathbf{m}}^{\mathbf{ec}} t^\circ$ ([Proposition 8.12](#)), which suffices to transport normalization results from DiLL to the calculus. Then if s° reduces in any way to a normal net t° then $s \xrightarrow{\mathbf{g}^*} t$ ([Theorem 8.16](#)), which suffices to show confluence of both reductions ([Corollary 8.17](#) for \mathbf{g} and [Theorem 8.6](#) for \mathbf{b} from it).

1.2 Notation

$\mathcal{M}_{\text{fin}}(X)$ is the set of finite multisets over X , i.e. functions $A : X \rightarrow \mathbb{N}$ with support $|A| < \omega$ finite. Depending on the context multisets will be presented either in additive (multiset union is $A + B$) or in multiplicative (AB) notation. In any case $\sum_{a \in A} D_a$ stands for a sum with multiplicities, i.e. $\sum_{a \in |A|} A(a) \cdot D_a$. For example cardinality is $\#A = \sum_{a \in A} 1$. If we need to denote a particular multiset, we do so by enclosing the elements in brackets, as in $[a, a, b, c, c]$.

Given a sequence $\vec{a} = (a_1, \dots, a_n)$, the sequence $(a_1, \dots, \hat{a}_i, \dots, a_n)$ denotes the sequence obtained from \vec{a} by skipping a_i .

The disjoint union of two sets A and B is denoted by $A + B$, with canonical injections given by $a \mapsto a.0 \in A + B$ for $a \in A$ and $b \mapsto b.1 \in A + B$ for $b \in B$.

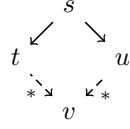
Basic notions of rewriting theory. An abstract reduction system is a set S together with a relation (the *reduction*) $\rightarrow \subseteq S \times S$. Composition of relations is denoted by juxtaposition, so that for example $\rightarrow \rightarrow$ is a two step reduction. We define the following derived relations:

- \leftarrow for the transpose of \rightarrow ;
- $\overrightarrow{\equiv}$, $\overrightarrow{+}$, $\overrightarrow{*}$ for its reflexive, transitive and reflexive-transitive closures;
- \equiv for its generated equivalence relation, i.e. its transitive, reflexive and symmetric closure $(\leftarrow \cup \rightarrow)^*$.

We say that \rightarrow is

- strongly confluent if $\leftarrow \rightarrow \subseteq \overrightarrow{\equiv} \overleftarrow{\equiv}$;
- locally confluent if $\leftarrow \rightarrow \subseteq \overrightarrow{*} \overleftarrow{*}$;
- confluent if $\overrightarrow{*}$ is strongly confluent.

Such conditions are usually drawn by a diagram. For example local confluence is depicted by stating that for term s, t, u :



The dashed lines indicate that such reduction exist, together with their target v .

A normal form is an element of S which is not in the left projection of \rightarrow , i.e. such that no further reduction is possible from it. We write $u \twoheadrightarrow v$ if $u \xrightarrow{*} v$ and moreover v is normal. If for a given u such v exists and is unique, we write $\text{NF}(u) := v$. We will then say that

- u is weakly normalizing, or $u \in \text{WN}$, if $\exists v \mid u \twoheadrightarrow v$;
- u is strongly normalizing, or $u \in \text{SN}$, if there is no infinite reduction $u \rightarrow u_1 \rightarrow \dots \rightarrow u_k \rightarrow \dots$;
- \rightarrow is weakly normalizing, or WN (resp. strongly normalizing, or SN) if every object of S is.

Said in other words, \rightarrow is SN if \leftarrow is well founded. For a strongly normalizing system, we will thus be able to use well founded induction (implicitly with respect to the reduction). If we can infer a property on t from the same property for all t' with $t \rightarrow t'$, then the property holds for all terms.

In Section 4.1 we will discuss more advanced notions of rewrite theory, dealing with reduction modulo equivalence, presenting the results we need.

Part I

Tools, Questions and Aims

Chapter 2

Nets

In this chapter we will outline the absolute protagonists of this work: **nets**. Starting from the presentation of proof nets in [Gir87] a wealth of redefinitions and extensions were made based on them [Dan90, Reg92, Gir91b, Gir96, Joi93, TdF00, Lau02, Pag06b, HvG03, ER06b] (and many others...). A generalization on which the nets we will use are based upon has been developed by Lafont in [Laf90], abstracting away the logical contents and concentrating on the computational one. These nets were later generalized to *multiport interaction nets* introduced by Mazza in [Maz06], introducing cells that could interact on more than one port, and *interaction structures*, thoroughly described by Vaux in [Vau07], in which differential interaction nets [ER06b] are described, as well as joins the previous two, introducing moreover the exponential boxes that are central to Part III. Also MALL proof nets as presented by Hughes and van Glabbeek in [HvG03], can be seen in this framework, though we will keep the slightly different definition of cut reduction there described (see 3.2.4).

We will therefore present here the formal framework in which subsequent chapters will move. We will strive to give a unifying framework in which all relatives of proof nets can be described. The definitions are mainly based on Vaux' ones in [Vau07]. Our main difference and contribution is the definition in all variants (interaction nets, sums, boxes and boxes with sums) of *contexts*. Especially when dealing with boxes this can give a uniform approach, as the reductions and conversions get defined exactly as in λ -calculus, by context closure.

Also in Section 2.3 we present in the most general way possible the framework in which *semantical interpretation* of nets takes place, by defining what in general a set denotational semantics is.

2.1 Statics

We will concentrate here on the static description and definition of our objects.

2.1.1 Ports, Cells and Wires

Intuitively nets are defined as circuits composed of *cells* whose *ports* are connected by *wires*. Cells come from sorts indicated by symbols in an alphabet,

and have a certain number (the *arity*) of input ports, one output *principal port* and a number (the *coarity*) of non-principal output ports¹.

Definition and Representation

A **signature** is given by an alphabet Σ and two functions $a, \bar{a} : \Sigma \rightarrow \mathbb{N}$ called **arity** and **coarity** respectively. The **degree** of a symbol α is $\deg(\alpha) := a(\alpha) + \bar{a}(\alpha) + 1$. For brevity, a signature will be denoted by its alphabet. Inclusion of signatures $\Sigma \subseteq \Sigma'$ is naturally given by the inclusion of the alphabets and by $a_{\Sigma'}|_{\Sigma} = a_{\Sigma}$ and $\bar{a}_{\Sigma'}|_{\Sigma} = \bar{a}_{\Sigma}$.

Let us fix a countable set, the set \mathcal{P} of **ports**. A **cell** c in Σ over \mathcal{P} is an tuple

$$c = (\alpha, p_0, \dots, p_k)$$

where $\alpha \in \Sigma$, $k = \deg(\alpha)$ and $p_0, \dots, p_{\deg(\alpha)} \in \mathcal{P}$ are distinct ports. We write

$$\begin{array}{ll} \sigma(c) := \alpha, & \text{its **symbol**,} \\ a(c) := a(\alpha), \bar{a}(c) := \bar{a}(\alpha), \deg(c) := \deg(\alpha), & \text{its arity, coarity and degree,} \\ \mathfrak{p}(c) := \{p_0, \dots, p_{\deg(\alpha)}\} & \text{its ports,} \\ \text{actv}(c) := (p_0, \dots, p_{\bar{a}(\alpha)}), & \text{its **active ports**,} \\ \text{pasv}(c) := (p_{\bar{a}(\alpha)+1}, \dots, p_{\deg(\alpha)}), & \text{its **passive ports**.} \end{array}$$

We denote by c_i with $0 \leq i \leq \bar{a}(c)$ the i -th active port, and by $c_{[i]}$ with $1 \leq i \leq a(c)$ the i -th passive port. The active port c_0 , always present, is called **principal port**, the others are called **auxiliary**.

A **net** μ on a signature Σ is given by a quadruple

$$(\mathfrak{p}(\mu), \mathfrak{c}(\mu), \mathfrak{pw}(\mu), \text{dl}(\mu)).$$

Ports: $\mathfrak{p}(\mu)$, the ports of μ , is a finite set in \mathcal{P} .

Cells: $\mathfrak{c}(\mu)$ is a set of cells with $\mathfrak{p}(c) \subseteq \mathfrak{p}(\mu)$ and disjoint two by two, i.e. if $a \neq b \in \mathfrak{c}(\mu)$ then $\mathfrak{p}(a) \cap \mathfrak{p}(b) = \emptyset$.

Proper wires: $\mathfrak{pw}(\mu)$, the set of **proper wires** of μ , is a partition of $\mathfrak{p}(\mu)$ into sets of cardinality 2, i.e. each port lies exactly in one proper wire.

Deadlocks: $\text{dl}(\mu) \in \mathbb{N}$ is the number of **deadlocks** of μ .

The **bounded** and **free ports** of μ are respectively the sets

$$\mathfrak{bp}(\mu) := \bigcup_{c \in \mathfrak{c}(\mu)} \mathfrak{p}(c), \quad \mathfrak{fp}(\mu) := \mathfrak{p}(\mu) \setminus \mathfrak{bp}(\mu).$$

It may be noted that $\mathfrak{p}(\mu)$ is completely determined by $\mathfrak{c}(\mu)$ and $\mathfrak{fp}(\mu)$, so that we may equivalently define these two sets and infer $\mathfrak{p}(\mu)$ from them. The **wires** of μ are $\mathfrak{w}(\mu) := \mathfrak{pw}(\mu) \cup \text{dl}(\mu)$ (where $n = \{0, \dots, n-1\}$, so that we are saying that wires are the proper ones plus dl distinguished ones called deadlocks). The set of **internal wires** is $\mathfrak{iw}(\mu) := \{e \in \mathfrak{pw}(\mu) \mid e \subseteq \mathfrak{bp}(\mu)\}$, i.e. the ones connecting bounded ports, while the **external** ones $\mathfrak{ew}(\mu)$ are the proper and non internal

¹There is no real distinction between the principal port and the other output ports, other than a naming convenience when it comes to our main use of multiple output ports: exponential boxes of linear logic.

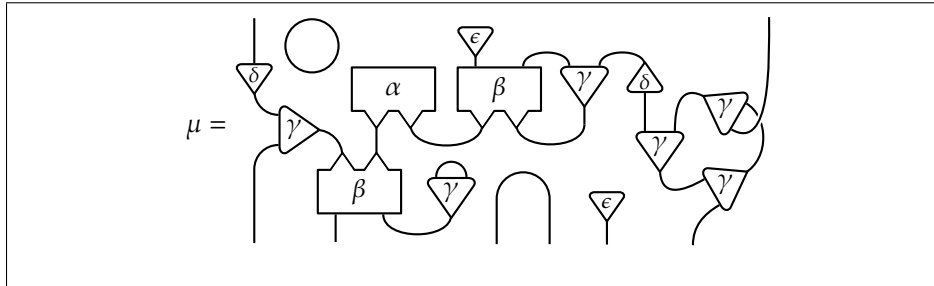


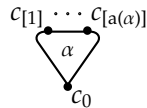
Figure 2.1: An example of net.

ones, thus the ones over the free ports. An external wire is said to be **dormant** if it does not link a passive port (these are the wires that can become cuts when the net is plugged in a context).

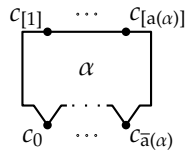
We adopt the idea of not giving a homogeneous definition for proper wires and deadlocks (as opposed to other treatments of the subject like [Maz06, Vau07]) as the latter play practically no role in our work. In such a case in many ways it is better to set them apart. Most notably, we thus have that ports are either free or belonging to a cell.

The above precise definition we be extensively used when proofs are carried out. However it somewhat hides the intuitive idea of net. So we use the following graphical representation, which bijectively characterizes nets.

Cells that have a single active port (also called **unicells**) as triangles containing their symbol where the active port is a vertex and the passive ones are placed on the opposite side left to right (relative to the vertex).



Cells that have more than one active port can be drawn with rectangles, with angles protruding from one side on whose vertexes active ports are placed, while passive ones are on the opposite side (left to right relative to the active side).



We will not however use much this notation, as the **multicells** (i.e. cells that are not unicells) that we will employ will only be boxes, which use a particular representation (see Section 2.1.3).

Wires, as their name suggests, are represented by edges between ports, and deadlocks by $dl(\mu)$ circles somewhere in the net. As each port is contained in exactly one wire, we can spot them as the extremities of wires, and we usually do not have to mark them as we did in the above drawings. Free ports, in particular, appear as extremities of dangling wires. So for example the net in Figure 2.1 has 8 free ports, 1 deadlock, is built on the signature $\Sigma = \{\alpha, \beta, \gamma, \delta, \epsilon\}$ where

α and β are the only multicell symbols with

$$a(\alpha) = 0, \quad \bar{a}(\alpha) = 1, \quad a(\beta) = 2, \quad \bar{a}(\beta) = 1,$$

while for unicell symbols

$$a(\delta) = 1, \quad a(\gamma) = 2, \quad a(\epsilon) = 0.$$

As can be seen in the graphical representation we do not give names to ports. Free ports can be identified by their position on the border, while we really do not want to make differences about what names are given to bounded ports, much like is done for bounded variables in λ -calculus. So the following notion is the equivalent of α -equivalence for nets, in order to abstract away the actual names of bounded ports.

Given two nets λ and μ with $dl(\lambda) = dl(\mu)$ an α -**conversion** from λ to μ is a bijection $\phi: \mathfrak{p}(\lambda) \rightarrow \mathfrak{p}(\mu)$ such that

- $\forall p \in \mathfrak{fp}(\lambda) : \phi(p) = p$, so that $\mathfrak{fp}(\lambda) = \mathfrak{fp}(\mu)$,
- $\forall (\alpha, p_0, \dots, p_k) \in \mathfrak{c}(\lambda) : (\alpha, \phi(p_0), \dots, \phi(p_k)) \in \mathfrak{c}(\mu)$,
- $\forall \{p, q\} \in \mathfrak{pw}(\lambda) : \{\phi(p), \phi(q)\} \in \mathfrak{pw}(\mu)$.

λ and μ are then said to be α -**equivalent**. From now on all we say has to be considered up to α -equivalence. It is not hard (and will not be shown explicitly) to see that all definitions and results are valid under such equivalence.

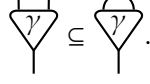
The set of (α -equivalence classes of) nets over a signature Σ is denoted by \mathbf{N}_Σ (dropping the subscript if it is clear from the context). There is a unique net with $\mathfrak{c}(\epsilon) = \emptyset$, $\mathfrak{fp}(\epsilon) = \emptyset$ and $dl(\epsilon) = 0$ (so that $\mathfrak{w}(\epsilon) = \emptyset$ also): we call ϵ the **empty net**.

Given a proper wire e we write $e(p) = q$ if $e = \{p, q\}$ (so $e^2(p) = p$). Given a net μ and a port p in it, we denote by p_μ^\perp (**dual of p in μ**) the port $e(p)$ where e is the unique wire in μ containing p . We drop the μ in subscript if it is clear from the context. Rather than defining $\mathfrak{pw}(\mu)$, we can equivalently define on $\mathfrak{p}(\mu)$ the involutive function $(-)_\mu^\perp$ without fixpoints, and recover from it $\mathfrak{pw}(\mu)$.

The set $\mathfrak{cw}(\mu)$ of **cuts** is the set of internal wires $\{p, q\}$ such that both p and q are active ports of different cells. The set $\mathfrak{aw}(\mu)$ of **axioms** is the set of wires $\{p, q\}$ with both p and q non-active (but not necessarily passive, i.e. bounded). In the graphical representation, cuts are wires between pointed parts of the cells, while axioms are between flat joints and/or free ports.

A **subnet** $\lambda \subseteq \mu$ is a net λ such that $\mathfrak{c}(\lambda) \subseteq \mathfrak{c}(\mu)$, $\mathfrak{iw}(\lambda) \subseteq \mathfrak{iw}(\mu)$ and $dl(\lambda) \leq dl(\mu)$. Notably, a subnet may have free ports that are not present in the main net, not even as bounded ports. We consider two subnets equal if they differ only by a renaming of free ports. In general we *do not* consider them up to α -equivalence also, as we want to be able to tell where exactly a subnet lies in a representant μ of an α -equivalence class. The set of subnets of a net μ is denoted by $\wp(\mu)$. There is a canonical injection from $\mathfrak{c}(\mu)$ into $\wp(\mu)$: to each cell c it associates the unique subnet with only c as cells, with no internal wires, no deadlocks and exactly $\deg(c)$ free ports. We then can regard every cell as a subnet also. There is also an injection $\mathfrak{cw}(\mu) \rightarrow \wp(\mu)$ that associates to each cut $e = \{c_i, d_j\}$ the smallest deadlock-free subnet containing c , d and e , denoted by $\mu \upharpoonright_e$.

One asks that only internal wires be inside the main net be able to say:



There is no way of finding the two top dangling wires in the right net, though intuition rightly tells that the left is indeed a subnet of the right.

A **directed proper wire** $d \in \text{dpw}(\mu)$ is a pair (p, q) with $\{p, q\} \in \text{pw}(\mu)$. The dual d^\perp is defined as the transpose (q, p) , the source and target by usual projections $s(d) := p$ and $t(d) := q$.

The **directed wire graph** $\vec{\mathcal{G}}(\mu)$ associated to a net μ is the one that has

- $\text{fp}(\mu) + \text{c}(\mu) + \text{dl}(\mu)$ as nodes;
- $\text{dpw}(\mu) + \text{dl}(\mu)$ as arrows;
- the following as source and target functions

$$s_{\vec{\mathcal{G}}}(d) := \begin{cases} d & \text{if } d \in \text{dl}(\mu), \\ s(d) & \text{if } s(d) \in \text{fp}(\mu), \\ c & \text{if } s(d) \in \text{p}(c) \text{ with } c \in \text{c}(\mu), \end{cases}$$

$$t_{\vec{\mathcal{G}}}(d) := \begin{cases} d & \text{if } d \in \text{dl}(\mu), \\ t(d) & \text{if } t(d) \in \text{fp}(\mu), \\ c & \text{if } t(d) \in \text{p}(c) \text{ with } c \in \text{c}(\mu). \end{cases}$$

In other words, one adds loops for deadlocks, and collapses all ports of a cell to a single node. An undirected version $\mathcal{G}(\mu)$ of the above definition can be easily derived. **Paths** in μ are the edges $d_1 \dots d_n$ of the free category generated by $\vec{\mathcal{G}}(\mu)$. By abuse of notation we will write

- $d_i \in \phi$ for every i ;
- $e \in \phi$ if e is an undirected wire $\{p, q\}$ such that either $(p, q) \in \phi$ or $(q, p) \in \phi$;
- $p \in \phi$ if p is a port with $p \in e \in \phi$;
- $c \in \phi$ if c is a cell with one of its ports $p \in \text{p}(c)$ with $p \in \phi$.

A path ϕ is said to be **elementary** if it is not empty, no undirected wire is repeated (i.e. if $i \neq j$ then neither $d_i = d_j$ nor $d_i^\perp = d_j$), and no three different ports of the same cell c are in ϕ (so that the path does not intersect itself, though it may still cycle). The dual of a path $\phi = d_1 \dots d_n$ is its reversal, i.e. $\phi^\perp := d_n^\perp \dots d_1^\perp$. A non-empty path ϕ is a cycle if

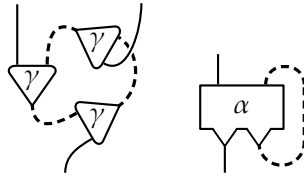
$$s_{\vec{\mathcal{G}}}(\phi) = s_{\vec{\mathcal{G}}}(e_1) = t_{\vec{\mathcal{G}}}(e_n) = t_{\vec{\mathcal{G}}}(\phi).$$

A directed wire d is **upward** if $s(d_i)$ is non active and $t(d_i)$ is active. We may refer to d in this case with the notation $\uparrow d$. Clearly d is **downward** (denoted by $\downarrow d$) if ϕ^\perp is upward. A path is upward or downward if all its wires are respectively upward or downward.

An elementary path ϕ **bounces** on a cell c if two auxiliary ports of c appear both in ϕ . This by elementarity condition means that the path enters through one of the two and exits immediately after from the second. An elementary path is **straight** if it never bounces.

An (elementary) path in $\mathcal{G}(\mu)$ is obtained from a directed path by forgetting the directions and quotienting by path reversal. Bouncing and being straight are properties that can also apply to undirected paths, while upward and downward are not.

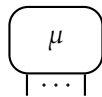
A deadlock loop, or a cycle ϕ that is upward or downward is called **vicious cycle**. Vicious cycles made only of unicells are parts of the net that cannot participate in a non-generalized reduction (see Section 2.2). The following net on the left is the smallest subnet of the one presented in Figure 2.1 that contains a vicious cycle. The one on the right is an example of vicious cycle consisting of only one wire.



Glueing and Contexts

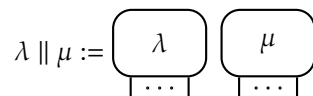
Many of the following definitions and results are usually quite obvious when interpreted in the graphical representation. So we first give an intuitive idea of definitions and facts that are used later. If one feels content with it and prefers not to delve too much in splitting hairs he may safely read just the next part and jump over the following one, that gives the rigorous definitions and proofs.

Ideas and Properties In the following, we will graphically denote general nets as in



where external wires are drawn as dangling wires that may start from any of the sides.

The **juxtaposition** of two nets λ and μ is the result of putting them side by side



This is a commutative operation that has ε as neutral element and such that both nets are subnets of it.

An **interface** is just an ordering of some ports of the net. Giving two interfaces of the same length to two nets lets us define the **glueing** $\langle \lambda, I \mid J, \mu \rangle$ of the two nets along their interfaces I and J . The idea is that we join

the corresponding external wires of the two interfaces. This is obtained from juxtaposition by identifying the interfaces. So if

$$\lambda = \left(\text{box } \lambda \text{ with interface } I \right) \quad J \left(\text{box } \mu \text{ with interface } J \right) = \mu$$

then

$$\langle \lambda, I \mid J, \mu \rangle := \left(\text{box } \lambda \text{ and } \mu \text{ plugged together} \right)$$

We point out as a particular case of this operation the notions of **context** and **module**. A context $\omega[\]$ is just a net ω with an interface I_ω , called its inner interface or hole. The only difference is that we redefine $\text{fp}(\omega[\]) := \text{fp}(\omega) \setminus I_\omega$. In order to distinguish this notion of context from its generalization to boxes, we call such a context **linear** (as the hole is not inside a box). A module μ is a net together with a total interface I_μ , i.e. one that covers all free ports. We thus have the very natural idea of plugging a module μ inside a context $\omega[\]$ as $\omega[\mu] := \langle \omega, I_\omega \mid I_\mu, \mu \rangle$, if the two interfaces match. While the symmetric notion of glueing may seem enough, it is not so if we want to extend it to boxes.

We call a module **proper** if it has no axiom between ports of its interface. Their main property is that it is not possible to plug them into different contexts and obtain the same net. In fact there is the following very simple counterexample

$$\begin{aligned} \lambda &= \text{arc} & \omega_1[\] &= \text{two triangles } \delta_1, \delta_2 \text{ with a dot on } \delta_1 & \omega_2[\] &= \text{two triangles } \delta_1, \delta_2 \text{ with a dot on } \delta_2 \\ \omega_1[\lambda] &= \omega_2[\lambda] & &= \text{two triangles } \delta_1, \delta_2 \end{aligned}$$

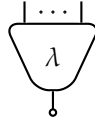
where we have $\omega_1[\lambda] = \mu = \omega_2[\lambda]$ but $\omega_1[\] \neq \omega_2[\]$. Though their underlying net is the same, their interfaces differ, as they themselves do extensionally (there is a net we can plug into them giving different results).

The most important property is then that if $\lambda \subseteq \mu$ is a proper module, there is a unique way of writing $\mu = \omega[\lambda]$ (Lemma 2.2). Given a local part of the net, there is an unambiguous way of speaking about what “the rest of the net” is. We may safely draw

$$\lambda \subseteq \mu \implies \mu = \left(\text{box } \omega[\] \text{ containing } \lambda \right)$$

This leads the way to neatly define rewriting as substituting a part (the redex) leaving the rest unchanged, i.e. context closure. This will be done in Subsection 2.2.

Unicells and Trees A tree is a net having a distinguished free port (the root), and inductively defined as being either a wire, or a unicell with its principal port connected to the root and with a tree on each of its passive ports. Graphically a tree is depicted as follows

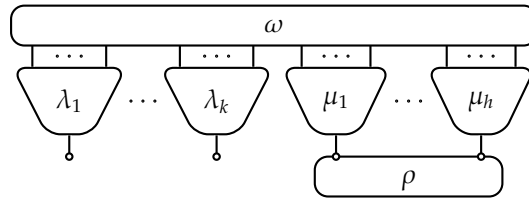


So the inductive rules can be depicted as

$$\lambda = \downarrow \text{ or } \quad ,$$

where $a(\alpha) = k$ and $\lambda_1, \dots, \lambda_k$ are trees.

Remark 2.1. Given a port p in a net λ with only unicells we can locate the maximal net $\mu \subseteq \lambda$ that is a tree rooted at p , by taking all wires and cells contained in all the maximal upward paths starting from p . Using this construction we can “factorize” every net made of unicells without vicious cycles in the following shape:



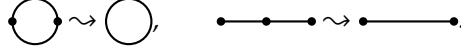
where $\lambda_1, \dots, \lambda_k, \mu_1, \dots, \mu_h$ are trees, while ω and ρ are **linkings**, i.e. nets with no cells and no deadlocks, made only by wires between free ports. If we impose that no μ_i is a single wire, we have (by Lemma 2.2, shown in the following) that this factorization is unique, and ω contains all the axioms of the net, while ρ all the cuts.

Definition and Proofs What we passed under silence in the previous part is how to define glueing using the rigorous definition of nets. The process recalls how composition of strategies in games is carried out (starting from Blass games [Bla92] and onwards): glued ports are like moves on the interface that need to be forgotten about.

A **pseudo-net** is the same as a net, where however there is another set of ports $\text{pp}(\mu)$ (the **pseudo-ports**), pw is a multiset rather than a set, and the condition on wires becomes that for each regular port there is exactly one wire containing it, while for each pseudo-port there are exactly two (which is the only case permitting repetition of wires). Intuitively a pseudo-net is a net where some wires have ports along them apart from the two extremities. A pseudo-net without pseudo-ports can be identified with a net (no repetition of wires is possible, so the multiset of proper wires can be identified with a set). We define a rewriting relation (**pseudo-port elimination**) on pseudo-nets: $\mu \rightsquigarrow \mu'$ if $c(\mu) = c(\mu')$, $\text{fp}(\mu) = \text{fp}(\mu')$, μ has a pseudo-port b in wires e and f and

- if e and f are parallel, i.e. $e = f$ (and so $e(b) = f(b) \in \text{pp}(\mu)$) then $\text{pp}(\mu') = \text{pp}(\mu) \setminus \{b, e(b)\}$, $\text{pw}(\mu') = \text{pw}(\mu) - [e, f]$ and $\text{dl}(\mu') = \text{dl}(\mu) + 1$; notice that in this case choosing b or $e(b)$ as redex gives the same reduct;
- otherwise $\text{pp}(\mu') = \text{pp}(\mu) \setminus \{b\}$, $\text{pw}(\mu') = \text{pw}(\mu) - [e, f] + [e \cup f \setminus \{b\}]$ and $\text{dl}(\mu') = \text{dl}(\mu)$.

This is to represent the two following graphical rewriting rules that just weld wires together:



This relation is easily seen to be strongly confluent and normalizing (the number of pseudo-ports decreases), so given a pseudo-net μ there is a unique net $\tilde{\mu}$ which is its “simplification”, the normal form with respect to \rightsquigarrow , with no pseudo-ports.

Given two nets λ and μ their **juxtaposition** $\lambda \parallel \mu$ is defined by the disjoint union of all the components of the nets (and thus sum of deadlocks). Clearly $\lambda \parallel \mu = \mu \parallel \lambda$, $\lambda \subseteq \lambda \parallel \mu$, $\mu \parallel \varepsilon = \mu$.

An **interface** $I = \vec{p}$ of a net λ is a non repeating sequence in $\text{fp}(\lambda)$. An interface is called **total** if it covers all $\text{fp}(\lambda)$. Interfaces are, as their name suggests, just a way to use the net in a standard way, i.e. know what ports are to be plugged where. Given two nets λ and μ with interfaces $I = \vec{p}$ and $J = \vec{q}$ with $\#I = \#J$, the **glueing** of λ and μ along I and J is defined as $\langle \lambda, I \mid J, \mu \rangle := \tilde{\lambda}$ where $\tilde{\lambda}$ is the pseudo-net obtained from $\lambda \parallel \mu$ by identifying p_i with q_i in the two interfaces, i.e. $\tilde{\lambda} = \lambda \parallel \mu /_{I \sim J}$ (quotient by the smallest equivalence relation equating I and J).

A **linear context** $\omega[]$ is a net ω together with a fixed partial interface $I_\omega^{[]}$ called **inner interface** or **hole**. For linear contexts we redefine the set $\text{fp}(\omega[]) := \text{fp}(\omega) \setminus I_\omega^{[]}$. As fp is redefined on contexts, a total interface of $\omega[]$ is not a total one of the underlying net ω . A **module** μ is a net together with a total interface I_μ on it. A module is said to be **proper** if for $p \in I_\mu$ we have $p_\mu^\perp \notin I_\mu$ (i.e. there is no axiom directly over the interface). Notice that cells seen as subnets can canonically get an interface (by the order of the ports), and are indeed proper modules. We define the plugging of a module in a context by $\omega[\mu] := \langle \omega, I_\omega^{[]} \mid I_\mu, \mu \rangle$ along the two interfaces (if they match), so that $\text{fp}(\omega[\mu]) = \text{fp}(\omega[])$. These notations are just a convenient way to denote a particular case of the symmetric glueing operation.

Lemma 2.2. *Given a proper module λ with $\lambda \subseteq \mu$, then there is a unique linear context $\omega[]$ such that $\omega[\lambda] = \mu$.*

Proof. Uniqueness is an application of [Vau07, Lemma 1.12] (there is no difficulty in adapting the result to our very slightly different syntax). For existence we build $\omega[]$ as follows. If $I_\lambda = (p_1, \dots, p_n)$ by hypothesis $(I_\lambda)_\lambda^\perp \subseteq \text{bp}(\lambda) \subseteq \text{bp}(\mu)$. Let:

- $\text{c}(\omega[]) := \text{c}(\mu) \setminus \text{c}(\lambda)$;
- $\text{fp}(\omega[]) := \text{fp}(\mu)$; $I_\omega^{[]} := (\bar{p}_1, \dots, \bar{p}_n)$ with \bar{p}_i “fresh” ports;
- $\text{pw}(\omega[]) := \{ f \in \text{pw}(\mu) \mid f \cap \text{bp}(\lambda) = \emptyset \} \cup$
 $\{ \{q, \bar{p}\} \mid p \in I_\lambda, q \in \text{p}(\omega[]), \{q, p_\lambda^\perp\} \in \text{pw}(\mu) \} \cup$
 $\{ \{\bar{q}, \bar{p}\} \mid q, p \in I_\lambda, \{q_\lambda^\perp, p_\lambda^\perp\} \in \text{pw}(\mu) \} ;$

- $\text{dl}(\omega[\]) = \text{dl}(\mu) - \text{dl}(\lambda)$.

Now it is clear from the definitions that $\text{c}(\omega[\lambda]) = \text{c}(\mu)$ and $\text{fp}(\omega[\lambda]) = \text{fp}(\mu)$. For every $q \in \mathfrak{p}(\mu)$ we show that $q_{\omega[\lambda]}^\perp = q_\mu^\perp$. Three cases are possible:

- $q, q_\mu^\perp \notin \text{bp}(\lambda)$, then $\{q, q_\mu^\perp\} \in \text{iw}(\omega[\]) \subseteq \text{pw}(\omega[\lambda])$;
- $q \in \text{bp}(\lambda)$ and $q_\mu^\perp \notin \text{bp}(\lambda)$ (or similarly viceversa) implies that $p := q_\lambda^\perp \in I_\lambda$, and $q_\mu^\perp \in \mathfrak{p}(\omega)$, so in $\omega \parallel \lambda$ we have the two wires $\{q_\mu^\perp, \bar{p}\}$ and $\{p, q\}$ which in $\omega \parallel \lambda /_{p_i \sim \bar{p}_i}$ reduce by \rightsquigarrow to $\{q_\mu^\perp, q\} \in \text{pw}(\omega[\lambda])$;
- $q, q_\mu^\perp \in \text{bp}(\lambda)$: if $\{q, q_\mu^\perp\} \in \text{iw}(\lambda)$ then such wire survives in $\omega[\lambda]$; otherwise by definition of subnet both $p := q_\lambda^\perp, p' := (q_\mu^\perp)_\lambda^\perp \in I_\lambda$ and therefore in $\omega \parallel \lambda$ we have $\{q, p\}, \{\bar{p}, \bar{p}'\}, \{p', q_\mu^\perp\}$ that simplify to $\{q, q_\mu^\perp\} \in \text{pw}(\omega[\lambda])$.

As a last thing, we also clearly have $\text{dl}(\omega[\lambda]) = \text{dl}(\mu)$. \square

Typing

Following [Vau07], a **type system** for a signature Σ is given by

- a set of **types** \mathcal{T} with an involutive negation $(-)^{\perp} : \mathcal{T} \rightarrow \mathcal{T}$ (i.e. such that $\tau^{\perp\perp} = \tau$);
- a relation $\vDash_{\alpha} \subseteq \mathcal{T}^{\text{deg}(\alpha)}$ for each $\alpha \in \mathcal{T}$, represented as

$$A_1, \dots, A_{\mathfrak{a}(\alpha)} \vDash_{\alpha} B_0, \dots, B_{\bar{\mathfrak{a}}(\alpha)}.$$

In the following examples we will use the signature for multiplicative linear logic, MLL. All symbols of such signature are unicell ones, so we can define only the arity:

$$\begin{array}{ll} \text{connectives:} & \otimes, \wp, \quad \mathfrak{a}(\otimes) = \mathfrak{a}(\wp) = 2, \\ \text{units:} & 1, \perp, \quad \mathfrak{a}(1) = \mathfrak{a}(\perp) = 0. \end{array}$$

So, the usual type system for it is the one generated by

$$\mathcal{T}_{\text{MLL}} ::= \mathcal{V} \mid \mathcal{V}^{\perp} \mid 1 \mid \perp \mid \mathcal{T}_{\text{MLL}} \otimes \mathcal{T}_{\text{MLL}} \mid \mathcal{T}_{\text{MLL}} \wp \mathcal{T}_{\text{MLL}},$$

where \mathcal{V} is a countable set of type variables, with involutive negation defined on units and connectives by De Morgan's duality:

$$1^{\perp} := \perp, \quad (A \otimes B)^{\perp} := A^{\perp} \wp B^{\perp}.$$

Relations are quite obviously defined by

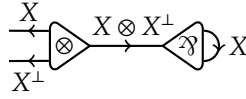
$$\begin{array}{ll} \vDash_1 := \{1\}, & \vDash_{\otimes} := \{(A, B, A \otimes B) \mid A, B \in \mathcal{T}_{\text{MLL}}\}, \\ \vDash_{\perp} := \{\perp\}, & \vDash_{\wp} := \{(A, B, A \wp B) \mid A, B \in \mathcal{T}_{\text{MLL}}\}. \end{array}$$

Given a net μ a **typing** on it is a function $\tau : \mathfrak{p}(\mu) \rightarrow \mathcal{T}$ such that

- $\forall c \in \text{c}(\mu) : \tau(c_{[1]}), \dots, \tau(c_{[\mathfrak{a}(c)]}) \vDash_{\sigma(c)} \tau(c_0)^{\perp}, \dots, \tau(c_{\bar{\mathfrak{a}}(c)})^{\perp}$;
- $\forall p \in \mathfrak{p}(\mu) : \tau(p^{\perp}) = \tau(p)^{\perp}$.

A **typed net** is just a net μ with a fixed typing τ_μ on it. The type of μ is then the *multiset* $\tau(\mu) := \tau_\mu \text{fp}(\mu)$. Such a multiset is as usually called **sequent**. The set of typed nets is denoted by $\mathbf{N}_{\Sigma, \mathcal{T}}$, again dropping the signature or the type system or both if they are clear from the context. Clearly we can equivalently give τ only on $\text{bp}(\mu)$ and extend it to $\text{fp}(\mu)$.

A typing τ induces a function $\tau: \text{dpw}(\mu) \rightarrow \mathcal{T}$ by $\tau(d) := \tau(\text{t}(d))$, such that $\tau(d^\perp) = \tau(d)^\perp$. In fact we can equivalently define such a function and retrieve back from it the typing of ports: given a port p contained in the unique wire $\{p, q\}$ we set $\tau(p) := \tau((q, p))$. This is the way how a typing is usually graphically represented. One chooses for each wire a direction (drawn with an arrow) and labels it with the corresponding type. Here is an example:



Remark. Note how again deadlocks are purposely left out of the discussion. Deadlocks are usually uninteresting from the point of view of typing, as they are always typable, no matter the system. Though in some contexts it may make sense assigning a type to a deadlock, it is of no interest here. Also in this way one can forget about them when proving results of subject reduction, as deadlocks pose no constraints on typing, and are inert from the point of view of reduction.

If \mathcal{T} is freely generated from a set of atoms, we say that a typed net μ is **η -expanded** if for all $\{p, q\} \in \text{aw}(\mu)$, $\tau_\mu(p)$ and $\tau_\mu(q)$ are atomic.

If \vDash_α are partial functions for all $\alpha \in \Sigma$ we say that \mathcal{T} is **axiomatic**. For example MLL (and in fact all type system we will use in this work) is axiomatic.

Lemma 2.3. *If \mathcal{T} is axiomatic and μ has no vicious cycles then typings on μ are completely determined by their values on axioms.*

Proof. For each non active port p let $n(p)$ be the maximal length of upward paths ϕ with $\text{s}(\phi) = p$, 0 if there is none (when p^\perp is non active also). This is always defined, as directed paths are non repeating and we move in a finite graph. What absence of vicious cycles gives is that for each p such that p^\perp is an active port of a cell c , for each $q \in \text{pasv}(c)$ we have $n(q) < n(p)$. If $n(p) = n(q)$ it means that we cannot add $\{p, p^\perp\}$ at the beginning of the longest upward path from q , and this happens only if either $\{p, p^\perp\}$ is already in it, or the cell c is already a target in it. In any case there is a cycle starting and ending in c . Therefore in a net with no vicious cycles we can easily reason by induction on $n(p)$.

Suppose now that we have two typings τ_1 and τ_2 that have the same values on axioms. Let us show that for every non active port p we have $\tau_1(p) = \tau_2(p)$, by induction on $n(p)$. If $n(p) = 0$ then p is in an axiom. Otherwise, $p^\perp = c_i$, and by induction hypothesis τ_1 and τ_2 are equal on all the passive ports \vec{q} of c . Then by definition $\tau_1(p^\perp) = \tau_2(p^\perp) = \vDash_{\sigma(c)}(\tau_1(\vec{q}))$. Now take any directed wire d . If it contains a non active port we are done. Otherwise it is a cut, and its type is the type of its target which is an active port, which as above is determined by the passive ports on the corresponding cell. \square

So a typing in an axiomatic type system on nets without vicious cycles is a dual-preserving function from directed axioms to types such that it can be extended to the whole net, where the conditions are dictated by the domains of ε_α and by equality on cuts.

If \mathcal{T} is freely generated from a set of variables \mathcal{V} , we call **atomic** the types in $\mathcal{V} \cup \mathcal{V}^\perp$. We say that a typed net μ is **η -expanded** if for all $\{p, q\} \in \mathbf{aw}(\mu)$, $\tau_\mu(p)$ and $\tau_\mu(q)$ are atomic.

Given an interface $I = \vec{p}$ on a typed net μ , then the type of the interface is naturally defined as $\tau_\mu(I) := (\tau_\mu(p_1), \dots, \tau_\mu(p_n))$. The type $\tau(\mu)$ is then redefined as the *sequence* $\tau_\mu(I_\mu)$, and the inner type of a typed context $\omega[\]$ is just $\tau^{[\]}(\omega) := \tau(I_\omega^{[\]})^\perp$ (the brackets in superscript differentiate this type from the usual one of interfaces, and the dual operation is carried out componentwise). The type of a typed linear context with a total interface is taken on such interface (therefore without taking into account the inner interface).

Lemma 2.4. *Given a typed linear context $\omega[\]$ and a typed module μ such that $\tau^{[\]}(\omega) = \tau(\mu)$ then $\omega[\mu]$ is a typed net with typing $\tau_{\omega[\mu]} := (\tau_\omega + \tau_\mu) \upharpoonright_{\mathbf{p}(\omega[\mu])}$ (so if $\omega[\]$ has a total interface the type of $\omega[\mu]$ is the same as the one of $\omega[\]$).*

Proof. One easily extends the definition of typing to pseudo-nets as a duality preserving function from directed wires to types, extended to ports as usual but only to regular (i.e. non-pseudo) ones, and adding the constraint that given a pseudo-port p contained in wires $\{r, p\}$ and $\{p, q\}$ then $\tau((r, p)) = \tau((p, q))$. If a pseudo-net is a net the definition is indeed the same.

The condition that the inner type of ω and the type of μ must match amounts then to $\tau_\omega + \tau_\mu$ (as functions on directed wires) being a typing on $\omega \parallel \mu / I_\omega^{[\]} \sim I_\mu$. The simplification reduction \rightsquigarrow is easily seen to preserve typing (by restriction to remaining wires). So $\tau_\omega + \tau_\mu$ restricted to the remaining ports is a typing on $\omega[\mu]$. \square

If τ is a typing on μ , and $\lambda \subseteq \mu$ then $\tau \upharpoonright_{\mathbf{bp}(\lambda)}$ defines a typing $\tau \upharpoonright_\lambda$ on λ , so that the above lemma is in fact an equivalence.

2.1.2 Sums

A **polynet** is a finite multiset of nets $\pi = [\mu_1, \dots, \mu_n]$ that share free ports, i.e. such that $\mathbf{fp}(\pi) := \mathbf{fp}(\mu_1) = \dots = \mathbf{fp}(\mu_n)$. $\mathbf{c}(\pi)$ and $\mathbf{w}(\pi)$ are defined as the sets of *occurrences* of the corresponding elements in the nets of π (by means of disjoint sums). Additive notation is used for these multisets, and simple nets μ are implicitly seen as the singleton nets $[\mu]$ (so that nets are canonically regarded as polynets also). When we want to explicitly distinguish a singleton polynet we may call it **simple**. We can and usually will write $\mu_1 + \dots + \mu_n := [\mu_1, \dots, \mu_n]$ and $k\mu = k[\mu]$. By definition the net $\mathbf{0} = []$ can be assigned any set of free ports.

The set of polynets over Σ is denoted by \mathbf{N}_Σ^+ . In fact $\mathbf{N}_\Sigma^+ \subseteq \mathbf{N}(\mathbf{N}_\Sigma)$ (i.e. the \mathbb{N} -module generated by \mathbf{N}_Σ).

Lemma 2.5. *The operations \mathbf{N} and \mathbf{N}^+ from signatures to sets are monotone increasing and continuous, i.e.*

$$\Sigma \subseteq \Sigma' \implies \mathbf{N}_\Sigma \subseteq \mathbf{N}_{\Sigma'}$$

and if Σ_i is an increasing sequence of alphabets then

$$\mathbf{N}_{\bigcup_{n \in \mathbb{N}} \Sigma_n} = \bigcup_{n \in \mathbb{N}} \mathbf{N}_{\Sigma_n},$$

and similarly for \mathbf{N}^+ .

Proof. The first claim is trivial from the definitions. For the second one, as $\mathbf{N}_{\Sigma_n} \subseteq \mathbf{N}_{\bigcup_{n \in \mathbb{N}} \Sigma_n}$ right-to-left inclusion is trivial.

Take a net $\mu \in \mathbf{N}_{\bigcup_{n \in \mathbb{N}} \Sigma_n}$ and consider

$$k := \max_{c \in c(\mu)} \min \{ h \mid \sigma(c) \in \Sigma_h \},$$

i.e. the maximal “index” of its cells with respect to $\bigcup_{n \in \mathbb{N}} \Sigma_n$, and we have that $\mu \in \mathbf{N}_{\Sigma_k}$. The proof for \mathbf{N}^+ is no different (it is paramount that the sums be finite). \square

The set $\wp(\pi)$ of subnets of π is defined as $\bigcup_{\lambda \in \pi} \wp(\lambda)$ (we do not consider “subpolynets”). Juxtaposition of nets is defined as

$$\pi \parallel \theta := [\lambda_i \parallel \mu_j \mid \lambda_i \in \pi, \mu_j \in \theta]$$

(counted with multiplicities).

Contexts

An **affine polycontext** is $\omega[] = k\delta[] + \pi$ (a linear context $\delta[]$ repeated k times plus a polynet π), where the occurrences of the internal interface are identified. Recall that the inner interface of contexts does not add up to free ports, so the inner interface $I_\omega^{[]} = I_\delta^{[]}$ is not shared by π . We call $\omega[]$ **linear** if $\pi = \mathbf{0}$. If $\omega[]$ is an affine polycontext and θ is a **polymodule**, (i.e. a polynet with a total interface) we can define $\omega[\theta]$ applying the generalized definition of glueing. In fact if $\omega[] = k\delta[] + \pi$ and if $\theta = \sum_i \mu_i$ with $\delta[]$ and μ_i are simple, then $\omega[\theta] = k \sum_i \delta[\mu_i] + \pi$. The nomenclature *linear* and *affine* is now clear: if $\omega[]$ is linear then $\omega[\mathbf{0}] = \mathbf{0}$ and $\omega[\theta_1 + \theta_2] = \omega[\theta_1] + \omega[\theta_2]$; if it is affine then $\omega[] - \omega[\mathbf{0}]$ is linear.

An affine polycontext $\omega[]$ is **monic** if $\omega[] = \delta[] + \pi$ with $\delta[]$ simple and linear (i.e. the coefficient of the net with the inner interface is 1). When plugging a net $\theta = \sum_i \mu_i$ in a monic context $\omega[] = \delta[] + \pi$ for each $p \in \mathfrak{p}(\omega[]) \setminus I_\omega^{[]}$ we call **residuals** of p in $\omega[\theta] = \sum_i \delta[\mu_i] + \pi$ the set:

- $\text{res}(p) := \{p\}$ if $p \in \mathfrak{p}(\pi)$;
- otherwise, if $p \in \mathfrak{bp}(\delta) \setminus I_\delta^{[]}$, $\text{res}(p) := \{p_1, \dots, p_n\}$ where p_i are the different occurrences of p in all $\delta[\mu_i]$.

Similarly we can define residuals of the cells of $\omega[]$.

Lemma 2.6. *For each proper module $\lambda \in \wp(\pi)$ there is a unique monic affine polycontext $\omega[]$ such that $\omega[\lambda] = \pi$.*

Proof. There is a simple $\mu \in \pi$ with $\lambda \subseteq \mu$. An immediate application of Lemma 2.2 gives a simple linear context $\delta[]$ such that $\omega[] = \delta[] + \pi - \mu$ (which is defined as $\mu \in \pi$, so $[\mu] \leq \pi$ as multisets) is the looked for polycontext. \square

Given a cut e in π we generalize $\pi|_e$ to be $\mu|_e$ where μ is the (occurrence of) net containing e .

Typing

A **typed polynet** π is a multiset of typed nets $\mu_1 + \dots + \mu_n$ such that for all $p \in \text{fp}(\pi)$ then $\tau_{\mu_i}(p)$ is the same for all i . We refer by τ_π to the direct union of the various functions τ_{μ_i} on $\text{p}(\pi)$. Notice that $\mathbf{0}$ is a typed net of any type. There is no difficulty in extending all typing related definitions to polynets. Lemma 2.3 for axiomatic type systems directly applies to polynets, and also Lemma 2.4 seamlessly extends to typed affine polycontexts.

Lemma 2.7. *If $\omega[\] = k\delta[\] + \pi$ is a typed affine polycontext and θ is a typed polymodule such that $\tau^{[1]}(\omega[\]) = \tau(\theta)$ then $\omega[\theta]$ is typed with*

$$\tau_{k\delta[\theta]+\pi} = k \sum_{\mu \in \theta} ((\tau_\omega \upharpoonright_\delta + \tau_\theta) \upharpoonright_{\delta[\mu]}) + \tau_\pi.$$

Notice that for a typed monic polycontext the residuals of a port p have thus the same type of p . Contrary to Lemma 2.4, the above one is not necessarily an equivalence. In order to show a brief counterexample let us sketch (a version of) the system MLL2 of second order MLL. It is obtained by adding two unary unicells \forall and \exists and the corresponding binding unary connectives $\forall X$ and $\exists X$ to the type grammar. The type system (which ceases to be axiomatic) is expanded with

$$\models_{\forall} = \{ (A, \forall X.A) \mid A \in \mathcal{T}, X \in \mathcal{V} \}, \quad \models_{\exists} = \{ (A[B/X], \exists X.A) \mid A \in \mathcal{T}, X \in \mathcal{V} \},$$

where $A[B/X]$ denotes as usual the formula A where B substitutes X . Then if

$$\theta := \begin{array}{c} \triangleleft \quad \triangleleft \\ \diagdown \quad \diagup \\ \otimes \\ \downarrow \end{array} + \begin{array}{c} \triangleleft \\ \downarrow \end{array}, \quad \omega[\] := \begin{array}{c} \circ \\ \triangleleft \\ \downarrow \end{array}.$$

we have θ untypable, while

$$\omega[\theta] = \begin{array}{c} \triangleleft \quad \triangleleft \\ \diagdown \quad \diagup \\ \otimes \\ \downarrow \\ \downarrow \quad \downarrow \\ \exists X.X \end{array} + \begin{array}{c} \triangleleft \\ \downarrow \\ \exists X.X \end{array}.$$

is typable.

Slices

Polynet closely resemble (and are a generalization) of another paradigm already studied in linear logic proofnets: *slice nets*. First hinted in [Gir87], then finely developed in [LTdF04] and [HvG03], their main application has been to model additives of linear logic. **Slice nets** are sets of nets sharing free ports. As is evident, the only difference with respect to polynets is the use of sets rather than multisets. In fact all the definitions and results shown for polynets seamlessly adapt to slice nets. This also follows from the fact that finite sets may be regarded as vectors in an R -module, where R is the unitary semiring $\{0, 1\}$ with

$1 + 1 = 1$, while the finite multisets we studied in the preceding sections are just vectors in an \mathbb{N} -module.

In particular we point out that juxtaposition has for slice nets the following definition:

$$\theta \parallel \phi := \{ \lambda \parallel \mu \mid \lambda \in \theta, \mu \in \phi \},$$

and that glueing is defined accordingly.

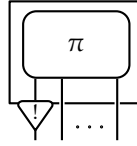
2.1.3 Boxes

Let \mathbf{N}_Σ^* (resp. \mathbf{N}_Σ^{+*}) denote *modules* (resp. *polymodules*) with at least one free port. Let Σ be a signature. Then let Σ^\square (resp. Σ^\boxplus) be the signature given by

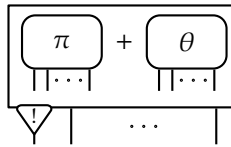
- the alphabet $\Sigma^\square := \Sigma + \mathbf{N}_\Sigma^*$ (resp. $\Sigma^\boxplus := \Sigma + \mathbf{N}_\Sigma^{+*}$); so whole nets become also symbols;
- arity and coarity are extensions to those of Σ with moreover, for $\pi \in \mathbf{N}_\Sigma^*$ (resp. \mathbf{N}_Σ^{+*}), $a(\pi) = 0$ and $\bar{a}(\pi) = \# I_\pi - 1$.

A cell B whose symbol is a net is called **box**, and its symbol $\sigma(B)$ is called its **contents**. Intuitively, Σ^\square is a signature for boxes containing simple nets, while Σ^\boxplus is one for boxes with whole sums inside them. Following the definition there is a natural bijection between the $\# I_\pi$ active ports of B and the interface I_π itself. We indicate by $(-)^B$ such bijection, or by $(-)^{\pi}$ if no confusion is possible. $b(\lambda)$ denotes the subset of $c(\lambda)$ that contains all boxes.

Boxes are represented by drawing their corresponding content... inside a box. For boxes of linear logic (the only ones we will use), the representation of a box of Σ^\square with symbol π is done by



where the port marked by the “fake” unicell with symbol ! is the principal one. The correspondence $(-)^B$ is therefore shown by the actual wires “exiting” the box. For boxes of Σ^\boxplus , we may use the same notation by implying that π is a sum, or, if a literal sum is present, by drawing the sum of nets inside without actually linking the ports outside with the ones inside, as in



where the correspondence $(-)^B$ is determined by position. In any case, we may draw ports also on other sides of the box, though they keep being active ones.

We now define over Σ the **box signatures** $\Sigma^!$ (**single-threaded**) and $\Sigma^{!+}$ (**multi-threaded**) by

$$\Sigma^! := \bigcup_{n \in \mathbb{N}} \Sigma^{\square n}, \quad \Sigma^{!+} := \bigcup_{n \in \mathbb{N}} \Sigma^{\boxplus n},$$

where $\Sigma^{\square n} := (\dots(\Sigma^{\square})^{\square}\dots)^{\square}$ repeated n times, and similarly for $\Sigma^{\boxplus n}$. As $\Sigma \subseteq \Sigma^{\square}$ and $\Sigma \subseteq \Sigma^{\boxplus}$ the above infinite unions are in fact limits of monotone sequences of sets. For this reason defining arity and coarity pose no problems.

By Lemma 2.5 we have that

$$\mathbf{N}_{\Sigma^!} = \bigcup_{n \in \mathbb{N}} \mathbf{N}_{\Sigma^{\square n}},$$

and similar equalities for other combinations between \mathbf{N}/\mathbf{N}^+ and $!/!+$. It makes therefore sense to define the **depth** $d(\pi)$ of a (simple) net over $\Sigma^!$ or $\Sigma^{!+}$ as the minimal integer such that $\pi \in \mathbf{N}_{\Sigma^{\square n}}$ (and analogously for all other combinations). Depth is in fact the maximal nesting of boxes in one another. It is the main parameter over which one reasons by induction when defining notions or proving propositions for nets with boxes.

Let F be any function from (poly)nets to finite sets (for example **pw**, **cw** or **b**). Then we inductively define:

$$F_0(\pi) := F(\pi), \quad F_{i+1}(\pi) := \sum_{B \in \mathbf{b}(\pi)} F_i(\sigma(B)).$$

It is immediate that $F_k(\pi) = \emptyset$ for $k > d(\pi)$, so it is natural to define $F_1(\pi) := \sum_{i=0}^{d(\pi)} F_i(\pi)$. So for example $c_1(\pi)$ is the set of *all* cells occurring in π , inside and outside boxes, or $\wp_1(\pi)$ is the set of all subnets occurring in π . For any element x of a net π (cell, wire, port, or even subnet and so on, i.e. $x \in F_1(\pi)$ for some F), then $x \in F_k(\pi)$ for a unique k . We say that k is the depth of x , writing $d(x) = k$. The **codepth** of x is $\text{cod}(x) := d(\pi) - d(x)$.

Contexts

Given a net μ over $(\Sigma + \Xi)^!$ (or $^{!+}$), we say that μ uses Ξ exactly once if there is a unique cell $c \in c_1(\mu)$ such that $\sigma(c) \in \Xi$. Given Σ , let Ξ (resp. Ξ^+) be the set of linear contexts (resp. affine polycontexts) with a total non-empty interface. Arities and coarities on Ξ and Ξ^+ are defined as in Σ^{\square} : given a context $\omega[]$ with total interface I_ω then $\mathbf{a}(\omega[]) = 0$ and $\bar{\mathbf{a}}(\omega[]) = \# I_\omega - 1$.

A **context** on $\Sigma^!$ is either a linear context on $\Sigma^!$, or a net on $(\Sigma + \Xi)^!$ that uses Ξ exactly once. A **polycontext** on $\Sigma^{!+}$ is either an affine polycontext on $\Sigma^!$, or a polynet on $(\Sigma + \Xi^+)^{!+}$ that uses Ξ^+ exactly once. The inner interface $I_\omega^{[]}$ is either trivially defined in the first cases, or else it is the inner interface of the element of Ξ/Ξ^+ uniquely appearing in $\omega[]$.

In fact it is the case that $\omega[]$ is a context on $\Sigma^!$ iff it is a linear context (in which case we say that the inner interface is at depth 0) or it is a simple net μ containing a single box B such that $\sigma(B)$ is inductively a context on $\Sigma^!$ (the inner interface is at the depth of it in $\sigma(B)$ plus one), and similarly for $\Sigma^{!+}$ contexts². This allows us to easily define $\omega[\pi]$ for matching $\omega[]$ and π by induction on the depth of the inner interface of $\omega[]$.

A polycontext on $\Sigma^{!+}$ is **monic** if it is either a monic affine one, or inductively if $\sigma(B)$ is monic, where B is the single box containing a context. A context on $\Sigma^!$

²This is the equivalent of the definition of context in λ -calculus, where it is asked that the hole be used exactly once. The more liberal syntax of nets asks however for such a long detour.

is always considered monic. Given a monic context $\omega[\]$ and a net θ , the residuals of $p \in \mathfrak{p}_!(\omega[\])$ in $\omega[\theta]$ have already been defined if $\omega[\]$ is affine, otherwise they are $\{p\}$ if p is not in the box B whose symbol $\sigma(B)[\]$ is a context, the residuals of p in $\sigma(B)[\theta]$ otherwise. Notice that extending similarly the definition to cells, contrary to what happens in affine contexts, the symbol of a cell may change in its residuals (namely boxes containing the hole change symbol).

Lemma 2.8. *Given a proper module λ such that $\lambda \in \wp_!(\pi)$ then there is a unique monic (poly)context $\omega[\]$ such that $\omega[\lambda] = \pi$.*

Proof. Straightforward induction on the depth of λ in π , by applying Lemma 2.6 in the base case. \square

Typing

A typing system \mathcal{T} with boxes is a typing system on a signature with boxes together with a relation $\vDash_{\square} \subseteq (\cup_n \mathcal{T}^n)^2$ such that

- $\vec{A} \vDash_{\square} \vec{B}$ implies $\# \vec{A} = \# \vec{B}$;
- for π symbol for a box (recall $\mathfrak{a}(\pi) = 0$, $\bar{\mathfrak{a}}(\pi) = \# I_{\pi} - 1$):

$$\vDash_{\pi} B_0, \dots, B_{\bar{\mathfrak{a}}(\pi)} \iff \exists \tau_{\pi} \text{ typing on } \pi \text{ such that} \\ \tau_{\pi}(I_{\pi}) \vDash_{\square} B_0, \dots, B_{\bar{\mathfrak{a}}(\pi)}.$$

Note that \vDash_{\square} can implement a boxing discipline, i.e. it can forbid the formation (at least in a typed setting) of boxes from nets that are not typed in a certain way. A box can be typed iff its content can be typed *and* the type of the content appears in $\pi_1(\vDash_{\square})$.

Given a typing τ on a polynet π in a typing system with boxes there is a typing $\tau_{\sigma}(B)$ for each $B \in \mathfrak{b}(\pi)$ witnessing the typing of B . By abuse of notation we extend τ on $\mathfrak{c}_!(\pi)$ with all these $\tau_{\sigma}(B)$, to obtain a typing on the whole net³.

A type system with boxes is axiomatic if \vDash_{α} is a partial function for every non-box symbol, and moreover \vDash_{\square} is a partial function. Lemma 2.3 extends to such a system.

Lemma 2.9. *In an axiomatic type system with boxes, if a net π has no vicious cycles at any depth, a typing is completely determined by its values on axioms.*

Proof. Easy adaptation of the one for Lemma 2.3, carried out by induction on the depth of the net. When considering a cell c we have to distinguish: if c is not a box then the types of its active ports are determined by those of the passive ones; if it is a box then they are determined by the typing inside the box, and inductive hypothesis kicks in. \square

The following lemma is an immediate consequence of the definitions and of Lemma 2.7.

Lemma 2.10. *Given a typed context $\omega[\]$ on $\Sigma^!$ or $\Sigma^{!+}$ and a typed module θ such that $\tau^{[\]}(\omega[\]) = \tau(\theta)$, then $\omega[\theta]$ is typed with*

³In fact such extension depends on the choice of witnesses, as there may be multiple ones to the same typing of a given box. We are practically redefining a typing to take into account also box contents.

- $\tau_{\omega[\theta]}$ as already defined for affine contexts if $\omega[\]$ is affine;
- otherwise $\tau_{\omega[\theta]}(p) := \tau_{\omega}(p)$ for $p \notin \mathbf{p}(\sigma(B)[\])$ where B is the box with the context, and inductively $\tau_{\omega[\theta]}(p) := \tau_{\sigma(B)[\]}(p)$ otherwise.

2.2 Dynamics

Objective of this section is to finally define what nets are all about and how one computes with them: we define the cut reduction relation.

Let Σ be a signature, whether with boxes or not. An **active pair** on Σ is a quadruple $\langle \alpha_i, \beta_j \rangle \in (\Sigma \times \mathbb{N})^2$ such that $0 \leq i \leq \bar{\alpha}(\alpha)$ and $0 \leq j \leq \bar{\alpha}(\beta)$. The set of active pairs is denoted by $\text{AP}(\Sigma)$. Given a (poly)net π every directed cut $e = (c_i, d_j)$ in it induces:

- the interfaced subnet $\pi|_e \in \wp(\pi)$ (or $\wp_1(\pi)$) which is the smallest containing c, d and e with free ports ordered by (c, d) ; formally it is defined by
 - $\mathbf{c}(\pi|_e) := \{c, d\}$;
 - $\mathbf{fp}(\pi|_e) := \{\bar{p} \mid \bar{p} \text{ fresh}, p \in \mathbf{bp}(\pi|_e) \setminus e\}$; they are put in the total interface $I_{\bar{\pi}_c}$ by order of appearance first in $\text{actv}(c)$ and then $\text{pasv}(c)$, $\text{actv}(d)$ and $\text{pasv}(d)$;
 - $\mathbf{pw}(\pi|_e) := \{e\} \cup \{\{p, \bar{p}\} \mid p \in \mathbf{bp}(\pi|_e) \setminus e\}$;
 - $\mathbf{dl}(\pi|_e) := 0$;
- a unique monic context $\omega(\pi, e)[\]$ (linear or affine or else depending on the system) such that $\omega(\pi, e)[\pi|_e] = \pi$, as $\pi|_e$ satisfies the hypotheses of Lemma 2.2 (or its equivalents Lemma 2.6 for sums and Lemma 2.8 for boxes);
- an active pair $\text{AP}(e) := \langle \sigma(c)_i, \sigma(d)_j \rangle$.

A **reduction system** \mathbf{r} over nets (resp. polynets) of a signature Σ is a partial function going from active pairs to modules (resp. polymodules) such that

- $\alpha_i \mathbf{r} \beta_j$ has $\text{deg}(\alpha) + \text{deg}(\beta) - 2$ free ports and has no cuts at depth 0⁴;
- $\alpha_i \mathbf{r} \beta_j$ is defined iff $\beta_j \mathbf{r} \alpha_i$ is;
- $\alpha_i \mathbf{r} \beta_j$ and $\beta_j \mathbf{r} \alpha_i$ have the same underlying net;
- if $I_{\alpha_i \mathbf{r} \beta_j} = (p_1, \dots, p_{\text{deg}(\alpha)-1}, q_1, \dots, q_{\text{deg}(\beta)-1})$ then

$$I_{\beta_j \mathbf{r} \alpha_i} = (q_1, \dots, q_{\text{deg}(\beta)-1}, p_1, \dots, p_{\text{deg}(\alpha)-1}).$$

Given a net π a **r-redex** in it is a subnet $\pi|_e$ for a directed cut e such that \mathbf{r} is defined on $\text{AP}(e)$.

Finally, having fixed a signature Σ and a reduction system \mathbf{r} on it, we define the reduction relation $\xrightarrow{\mathbf{r}}$ on nets by context closure, i.e. $\pi \xrightarrow{\mathbf{r}} \pi'$ iff there is a redex $\pi|_e \in \wp_1(\pi)$ and $\pi' = \omega[\mathbf{r}(\text{AP}(e))]$ where $\omega[\]$ is the unique monic context

⁴Usually one asks that the reducts be cut free. However this would break the definition of reduction for boxes (if we consider a net with a box with cuts to have cuts).

such that $\pi = \omega[\mu|_e]$. Given a port or a cell, its **residuals** in π' are its residuals with respect to $\omega[\]$ if it is in $\omega[\]$, and \emptyset otherwise.

Graphically reduction rules are represented by drawing the redexes on one side and the reducts on the other. For example MLL has the following reduction rules:

$$\begin{array}{ccc} \begin{array}{c} \text{---} \otimes \text{---} \\ \diagdown \quad \diagup \\ \text{---} \gamma \text{---} \end{array} & \xrightarrow{\mathfrak{m}} & \begin{array}{c} \text{---} \\ \diagdown \quad \diagup \\ \text{---} \end{array} \end{array} \quad \begin{array}{ccc} \begin{array}{c} \text{---} 1 \text{---} \\ \diagdown \quad \diagup \\ \text{---} 1 \text{---} \end{array} & \xrightarrow{\mathfrak{m}} & \varepsilon \end{array}$$

where \mathfrak{m} stands for *multiplicative reduction*, and is easily seen to be compatible with typing.

For example we have the following reduction.

$$\begin{array}{ccc} \begin{array}{c} \text{---} \otimes \text{---} \\ \diagdown \quad \diagup \\ \text{---} \gamma \text{---} \end{array} & \xrightarrow{\mathfrak{m}} & \begin{array}{c} \text{---} \\ \diagdown \quad \diagup \\ \text{---} \end{array} = \bigcirc \end{array}$$

Remark. Lafont's interaction nets are simple nets over a finite signature such that $\bar{a} \equiv 0$. Note that the reduction on nets with boxes also goes out of Mazza's multiport interaction nets paradigm: when we reduce a redex at depth greater than zero from the point of view of the zero depth net there is a cell (the box containing the cut) that changes (i.e. reduces) without interacting with any cell.

The reduction relation presented reduces, as expected, cuts and nothing else. The presentation of a reduction system is neat and very general, but at times other ways of rewriting nets is needed, where cuts are not central. We address such issue with the following definition.

A **generalized reduction system** \mathfrak{s} over a signature Σ is given by

- a set $R(\mathfrak{s})$ (the **s-redexes**) be a set of cut-free proper simple modules, such that no two redexes share the same underlying net; an **s-redex** in π is then a proper module $\mu \subseteq \pi$ such that $\mu \in R$ (up to isomorphism);
- a function denoted by \mathfrak{s} from $R(\mathfrak{s})$ to modules such that $\mathfrak{s}(\mu)$ has $\# I_\mu$ free ports and $\wp_0(\mathfrak{s}(\mu)) \cap R = \emptyset$.

One could ask some more properties from a generalized reduction system, like connectedness of redexes, or some degree of "simplicity". However we just need here a general definition, properties will be proved case by case in the future.

Again, $\xrightarrow{\mathfrak{s}}$ is defined by context closure: $\pi \xrightarrow{\mathfrak{s}} \pi'$ iff there is an **s-redex** $\mu \in \wp_1(\pi)$ and $\pi' = \omega[\mathfrak{s}(\mu)]$ where $\omega[\]$ is the unique context with $\pi = \omega[\mu]$.

Clearly a reduction system can be seen as a generalized one, by choosing one redex for each two symmetric active pairs.

Given two redexes λ and μ in π , they are said to be **orthogonal** if μ is a subnet of ω where $\omega[\]$ is the unique context with $\omega[\lambda] = \pi$. The main property of unicell nets with a standard reduction is that all redexes are orthogonal, as each cell has at most one cut and so is in at most one redex. Two overlapping redexes, i.e. non orthogonal, are called a **critical pair**.

Given a reduction system, the relations $\xrightarrow{\mathfrak{s}}$, $\xrightarrow{\mathfrak{s}}$, $\xrightarrow{\mathfrak{s}}$ and $\equiv_{\mathfrak{s}}$ mean respectively the reflexive-transitive, reflexive, transitive and reflexive-transitive-symmetric closures of $\xrightarrow{\mathfrak{s}}$. Given two reductions \mathfrak{r} and \mathfrak{s} with disjoint domains (i.e. sets of redexes), then \mathfrak{rs} is the union of the two functions. In fact we then have $\xrightarrow{\mathfrak{rs}} = \xrightarrow{\mathfrak{r}} \cup \xrightarrow{\mathfrak{s}}$.

2.2.1 Subject Reduction

A reduction system \mathbf{r} has the property of **subject reduction** with respect to a type system \mathcal{T} if for each active pair $\langle \alpha_i, \beta_j \rangle$ such that $\alpha_i \mathbf{r} \beta_j$ and for each choice of types \vec{A} and \vec{B} with

$$A_{\bar{\alpha}(\alpha)+1}, \dots, A_{\text{deg}(\alpha)} \vDash_{\alpha} A_0, \dots, A_{\bar{\alpha}(\alpha)}, \quad B_{\bar{\alpha}(\beta)+1}, \dots, B_{\text{deg}(\beta)} \vDash_{\beta} B_0, \dots, B_{\bar{\alpha}(\beta)}$$

and $A_i = B_j$ there is a typing τ on $\alpha_i \mathbf{r} \beta_j$ such that

$$\tau(\alpha_i \mathbf{r} \beta_j) = (A_0, \dots, \hat{A}_i, \dots, A_{\text{deg}(\alpha)}, B_0, \dots, \hat{B}_j, \dots, B_{\text{deg}(\beta)}).$$

For a generalized reduction systems, this extends to saying that for each $\mu \in \mathbf{R}(\mathbf{r})$ there is a function

$$(-)_{\mathbf{r}(\mu)}: \{ \tau \mid \tau \text{ typing on } \mu \} \rightarrow \{ \tau' \mid \tau' \text{ typing on } \mathbf{r}(\mu) \}$$

such that $\tau_{\mathbf{r}(\mu)}(\mathbf{r}(\mu)) = \tau(\mu)$.

Lemma 2.11. *If \mathbf{r} has subject reduction so does $\xrightarrow{\mathbf{r}}$, i.e. given a typed net π with $\pi \xrightarrow{\mathbf{r}} \pi'$, then there is a typing $\tau_{\pi'}$ such that for each port p in π and each of its residuals p' in π' we have that $\tau_{\pi'}(p') = \tau_{\pi}(p)$. In particular $\tau_{\pi'}(\pi') = \tau_{\pi}(\pi)$.*

Proof. This is a direct consequence of Lemma 2.10. If $\pi = \omega[\mu]$ and $\mu \in \mathbf{R}(\mathbf{r})$ then τ_{π} induces a typing on both $\omega[\]$ and μ . The latter induces by definition a typing of $\mathbf{r}(\mu)$ with type equal to $\tau_{\pi}(\mu)$. Therefore $\pi' = \omega[\mathbf{r}(\mu)]$ is typed, and the type of the residuals p' of a port p is $\tau_{\omega}(p) = \tau_{\pi}(p)$. \square

2.2.2 Structural Congruence

At times the structure of nets is in some way more “rigid” than it should, in the system we want to implement. In this case one needs a way to easily equate up to some equivalence different nets, in a way that is consistent with other definitions, like typing or reduction. Suppose fixed a type system \mathcal{T} and a reduction system \mathbf{r} . A **cell structural equivalence** \equiv is given by pairs $\lambda \equiv \mu$ of simple proper modules with equal number of free ports and such that both λ and μ are made of a single cell and no internal wire. The relation \equiv on nets then denotes the smallest equivalence relation closed by contexts, that is if $\pi \equiv \theta$ then $\omega[\pi] \equiv \omega[\theta]$ for any context $\omega[\]$. We now further ask that

- for each generating pair $\lambda \equiv \mu$ then each typing of λ induces one of μ and viceversa with same type on the interface;
- for each generating pair $\lambda \equiv \mu$ and each \mathbf{r} -redex ν such that $\lambda \subseteq \nu$ (which implies uniquely $\nu = \delta[\lambda]$) then $\delta[\mu]$ is a \mathbf{r} -redex such that $\mathbf{r}(\delta[\mu]) \equiv (\nu)$.

Note that as we required the generating pairs to be single cells, there are no critical peaks between congruence conversion and reduction, so we can check singularly each reduction rule separately, and such verification is generally very straightforward. This check is a very strict requirement which leaves out many equivalences that can be found in the literature, but which is really safe from the point of view of reduction. It implies that reduction is well defined on \equiv -classes, and that converting a net into an equivalent one does not change which

reductions we can do, and the length of each of them. So it is safe from every point of view to redefine nets to be in \mathbf{N}/\equiv (and \mathbf{N}/\cong).

We are very strict as we will use just two instances of structural equivalence which we define in the following and that fall in such formalism. Other “traditional” equivalences such as associativity of contraction are treated by means of generalized reductions in this work.

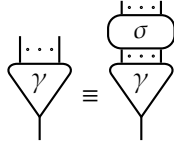
Commutativity. Every permutation $\sigma \in S_n$ induces naturally a simple module (also called σ) with interface $I_\sigma := (p_1, \dots, p_n, q_1, \dots, q_n)$, no cells, no deadlocks and

$$\text{pw}(\sigma) := \{ \{p_i, q_{\sigma(i)}\} \mid 1 \leq i \leq n \}.$$

As an example, here is a permutation of six elements and the corresponding module (p_i ports drawn above, q_i ports drawn below, both left to right):

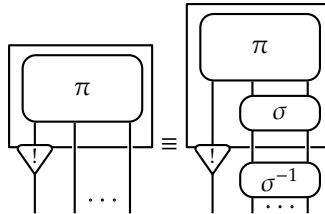
$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 1 & 3 & 5 & 6 & 4 \end{pmatrix} \mapsto \begin{array}{c} \text{---} \\ | \\ | \\ | \\ | \\ | \\ \text{---} \\ \sigma \end{array} = \begin{array}{c} \text{---} \\ | \\ | \\ | \\ | \\ | \\ \text{---} \\ \text{---} \\ | \\ | \\ \text{---} \end{array}$$

Given a set of unicast symbols $\Xi \subseteq \Sigma$, and if \equiv is a cell structural congruence generated by pairs



with $\gamma \in \Xi$ and $\sigma \in S_{a(\gamma)}$ we say that Ξ are commutative cells in \mathbf{N}_Σ/\equiv and \mathbf{N}_Σ/\cong . This can be reworded by saying that the passive ports of cells with symbols $\gamma \in \Xi$ are treated as sets, rather than as sequences, and that the type and reduction systems are well defined with respect to this.

Box port exchange. Another structural equivalence we use is a very natural one arising with boxes. If one inspects the definition, one may note that the symbols inside boxes are modules, thus are defined not only by the net, but also by the order of conclusions, which then corresponds to the order of auxiliary ports. The only thing that matters is however the correspondence between such auxiliary ports and the ports inside. So we equate nets with boxes by means of the following cell structural congruence.



where $\sigma \in S_{\#I_\pi - 1}$. Notice that by definition of glueing the above definition is valid also if π is a sum. Again, we can intuitively restate this as saying that the content of a box is a net with a distinguished port (but with the other ones unordered), and its auxiliary ports are, in a way, the set of the other free ports of the contents (and again unordered, though distinguishable).

2.2.3 Confluence

We here address confluence issues. We will restrict ourselves to a signature Σ with only unicells (so in particular no boxes). In other cases confluence must be rather addressed case by case. In our particular case the proof of such property will occupy the whole of Chapter 6.

The following is a very straightforward property of Lafont's interaction nets, following from the orthogonality of all redexes.

Proposition 2.12 (Lamont [Laf95]). *Any (non generalized) reduction \mathbf{r} on nets of a signature of unicells is strongly confluent.*

When moving on to polynets there comes a slight nuisance. Reducing a cut may give a non-simple polynet, and so it may duplicate or erase (if the result is 0) other redexes in the same simple addend. This should pose no problem, as in case of duplication the copies are completely independent. In order to tackle such problem we introduce a more parallel version of the reduction, closer to what seen in [ER06b, Vau07].

The **sum reduction** relation $\overset{\Sigma\mathbf{r}}{\rightrightarrows}$ on polynets is defined as

$$\pi \overset{\Sigma\mathbf{r}}{\rightrightarrows} \sigma \iff \pi = \sum_{i=1}^k \lambda_i, \sigma = \sum_{i=1}^k \mu_i, \forall i : \lambda_i \overset{\mathbf{r}}{\rightrightarrows} \mu_i.$$

We can restrict (but need not to) all λ_i s in the definition to be simple. Note that

$$\mathbf{r} \subseteq \overset{\Sigma\mathbf{r}}{\rightrightarrows} \subseteq \mathbf{r}^*. \quad (2.1)$$

The last inclusion follows from the independence of reduction in various addends, as

$$\begin{aligned} \forall i : \lambda_i \overset{\mathbf{r}}{\rightrightarrows} \mu_i \implies \lambda_1 + \dots + \lambda_k \overset{\mathbf{r}}{\rightrightarrows} \mu_1 + \lambda_2 + \dots + \lambda_k \overset{\mathbf{r}}{\rightrightarrows} \dots \overset{\mathbf{r}}{\rightrightarrows} \\ \mu_1 + \dots + \mu_{k-1} + \lambda_k \overset{\mathbf{r}}{\rightrightarrows} \mu_1 + \dots + \mu_k. \end{aligned}$$

Equation (2.1) assures that $(\overset{\Sigma\mathbf{r}}{\rightrightarrows})^* = \mathbf{r}^*$, as it implies

$$\mathbf{r}^* \subseteq (\overset{\Sigma\mathbf{r}}{\rightrightarrows})^* \subseteq (\mathbf{r}^*)^* = \mathbf{r}^*. \quad (2.2)$$

Lemma 2.13. *Let \mathbf{r} be a non generalized reduction on polynets of a signature of unicells. If λ is a simple net, $\lambda \overset{\mathbf{r}}{\rightrightarrows} \pi_1, \pi_2$ then there is σ with $\pi_1, \pi_2 \overset{\Sigma\mathbf{r}}{\rightrightarrows} \sigma$.*

$$\begin{array}{ccc} & & \pi_1 \\ & \nearrow^{\mathbf{r}} & \dashrightarrow^{\Sigma\mathbf{r}} \\ \lambda & & \sigma \\ & \searrow_{\mathbf{r}} & \dashrightarrow_{\Sigma\mathbf{r}} \\ & & \pi_2 \end{array}$$

Proof. This is an easy generalization of the result on interaction nets. Suppose the two reductions actually reduce different cuts c_1 and c_2 . The cases in which this does not happen are trivial. Suppose now $\mathbf{r}(\lambda \upharpoonright_{c_1}) = \sum \mu_i$ and $\mathbf{r}(\lambda \upharpoonright_{c_2}) = \sum \nu_j$. Then $c_1 \in \text{iw}(\omega(\lambda, c_2))$ (the unique context of the cut c_2), $c_2 \in \text{iw}(\omega(\lambda, c_1))$, so we can define residuals of c_1 in π_2 and of c_2 in π_1 .

So in $\pi_1 = \sum_i \omega(\lambda, c_1)[\mu_i]$ we can reduce each residual of c_2 and get

$$\sum_i \omega(\lambda, c_1)[\mu_i] \xrightarrow{\Sigma \mathfrak{r}} \sum_i \sum_j \omega(\omega(\lambda, c_1)[\mu_i], c_2)[\nu_j]. \quad (2.3)$$

By associativity of glueing

$$\omega(\omega(\lambda, c_1[\mu_i]), c_2)[\nu_j] = \omega(\omega(\lambda, c_2[\nu_j]), c_1)[\mu_i]$$

so (by commuting sums) the right member of (2.3) is symmetrically the result of reducing all residuals of c_1 in π_2 . \square

Notice that the above result covers also the case in which λ is erased, i.e. π_1 or π_2 (or both) are 0. Reflexivity of $\xrightarrow{\Sigma \mathfrak{r}}$ kicks in in such cases.

Corollary 2.14. *For any (non generalized) reduction \mathfrak{r} on polynets of a signature of unicells, $\xrightarrow{\Sigma \mathfrak{r}}$ is strongly confluent.*

Proof. This easily follows from the fact that if $\xrightarrow{\Sigma \mathfrak{r}}$ is compatible with sum, i.e. if $\lambda \xrightarrow{\Sigma \mathfrak{r}} \pi$ and $\mu \xrightarrow{\Sigma \mathfrak{r}} \sigma$ then $\lambda + \mu \xrightarrow{\Sigma \mathfrak{r}} \pi + \sigma$. If a term π forks by means of two sum reductions to two terms σ_1 and σ_2 by definition it means that each of its addends fork with $\xrightarrow{\mathfrak{r}}$, which by the above result can be joined with sum reduction, which in turn by compatibility can be put together to a join of σ_1 and σ_2 . \square

This corollary together with (2.2) easily gives confluence of $\xrightarrow{\mathfrak{r}}$, similarly to what is done in untyped λ -calculus and parallel reduction. More generally we have the following result.

Proposition 2.15. *Every reduction whose reflexive closure contains $\xrightarrow{\mathfrak{r}}$ and that is contained in $\xrightarrow{\Sigma \mathfrak{r}}$ is confluent.*

The above result comprises for example the *strict* sum reduction, i.e. the one that as sum reduction can reduce multiple addends, but strictly reduces at least one of them. This is the reduction defined in [ER06b, Vau07], which was done in order to recover strong confluence and to define it with non trivial coefficients. As we restrict ourselves to integer coefficients, and all results proved for such an atomic reduction easily apply also to the sum reduction, we decided rather to stick to the more intuitive idea of one step reduction.

2.3 Invariants

We will define in a very general way what a relational denotational semantics is. Our definition is a generalization of *experiments* for linear logic proof structures: introduced by Girard in [Gir87], they have been studied in [DdW94, TdF00, Pag06b]. We will define them as a particular case of typing. In this way we can rely on all the definitions and results we have already been through, the most important of which is compositionality of typing with respect to contexts and modules.

Though we will not deal with denotational semantics of exponential proof nets, we will anyway give the basic notions needed to define such semantics.

2.3.1 Denotational Semantics by Experiments

First of all we need to have at our disposal a slightly modified definition of typing. The first applies only to polynets and slice nets, the second only to signature with boxes.

A **selective type system** for polynets (resp. slice nets) is a type system here typed polynets (resp. slice nets) are defined as multisets (resp. sets) of untyped nets plus a single typed one. Formally, a typing τ_π of π becomes a pair (μ_i, τ_{μ_i}) where $\mu_i \in \pi$ and τ_{μ_i} is a typing of μ_i for some i (in fact the typings of a net become the disjoint union of the typings of their simple nets). Then if I_π is a total interface for π , $\tau(\pi) := \tau_{\mu_i}(I_\pi)$. Notice that in selective type systems $\mathbf{0}$ and \emptyset are not typable.

An **extended type system with boxes** is a type system \mathcal{T} with boxes where however the typing relation \vDash_\square associated with boxes is a relation between $\mathcal{M}_{\text{fin}}(\cup_{n \geq 1} \mathcal{T}^n)$ and $\cup_{n \geq 1} \mathcal{T}^n$. We recall that \mathcal{T} is the set of types. The conditions on the typing are changed to

- $[\vec{A}_1, \dots, \vec{A}_k] \vDash_\square \vec{B}$ implies $\# \vec{A}_i = \# \vec{B}$ for all i ;
- for π symbol for a box:

$$\vDash_\pi B_0, \dots, B_{\bar{a}(\pi)} \iff \exists [\tau_1, \dots, \tau_k] \text{ typings on } \pi \text{ such that} \\ [\tau_1(I_\pi), \dots, \tau_k(I_\pi)] \vDash_\square B_0, \dots, B_{\bar{a}(\pi)}.$$

One can refer to Section 2.1.3 to see the difference with the usual typing.

Extended type systems, apart from offering a common ground to the upcoming definition of set denotational semantics, can also be the base for defining intersection type systems on nets. Lemmas 2.7 and 2.10 are straightforwardly seen to be valid for selective or extended type systems, or both.

Let us fix a signature (with or without boxes) and a reduction system \mathfrak{r} on it. Take a class⁵ \mathcal{U} (the **universe of points**). A relational denotational semantics $\llbracket \cdot \rrbracket$ is an extended selective type system having \mathcal{U} as types, such that it has the subject reduction property and for each redex μ the function $(-)_\mathfrak{r}(\mu)$ is surjective, i.e. not only every typing on a redex induces a typing on its reduct with the same type, but also viceversa.

We recall that therefore a denotational semantics comes equipped with relations $\vDash_{[\alpha]} \subseteq \mathcal{U}^{\text{deg}(\alpha)}$ for each non-box symbol α , and a relation $\vDash_{[\square]}$ for boxes, where we use $\llbracket \cdot \rrbracket$ to distinguish from the usual notion of type system. In this setting we call a typing ϵ of a net π an **experiment on π** , and (if π has a total interface) the type $\epsilon(\pi) = \epsilon(I_\pi) \in \mathcal{U}^{\# I_\pi}$ **result** of the experiment. The interpretation of a module π is then

$$\llbracket \pi \rrbracket := \{ \epsilon(\pi) \mid \epsilon \text{ experiment on } \pi \} \subseteq \mathcal{U}^{\# I_\pi}.$$

A relational denotational semantics for a type system \mathcal{T} is one equipped with a function $\llbracket \cdot \rrbracket$ from \mathcal{T} to sets of points in \mathcal{U} . The meaning of the additional $|\cdot|$ notation will be made clear later. An experiment ϵ on a net π is then compatible with a typing τ_π on π if inductively (supposing ϵ is taken on $\mu \in \pi$):

- $\forall p \in \mathfrak{p}_0(\mu) : \epsilon(p) \in \llbracket \tau_\pi(p) \rrbracket$;

⁵We use a class in order to be able to interpret nets in the whole of **Rel**.

- $\forall B \in \mathbf{b}_0(\mu)$: the experiments $\mathbf{e}_1, \dots, \mathbf{e}_n$ associated by \mathbf{e} with B are compatible with $\tau_\pi \upharpoonright_{\sigma(B)}$.

The function $(-)\mathbf{r}(\mu)$ from experiments on the redex μ to those of its reduct is now required to preserve compatibility and to be surjective on compatible experiments.

The interpretation of a typed module π is then

$$\llbracket \pi \rrbracket := \{ \mathbf{e}(\pi) \mid \mathbf{e} \text{ experiment on } \pi \text{ compatible with } \tau_\pi \} \subseteq \prod_{p \in I_\pi} \llbracket \tau_\pi(p) \rrbracket.$$

The following lemma states that indeed our notion of denotational semantics is a denotational semantics in the usual sense, i.e. it is invariant under reduction.

Lemma 2.16. *If $\llbracket \cdot \rrbracket$ is a relational denotational semantics (for a type system \mathcal{T}) and $\pi \xrightarrow{\tau} \pi'$ then $\llbracket \pi \rrbracket = \llbracket \pi' \rrbracket$.*

Proof. Let $\pi = \omega[\mu]$ and $\pi' = \omega[\mathbf{r}(\mu)]$ with $\mu \in \mathbf{R}(\mathbf{r})$. If \mathbf{e} is an experiment on π , then by subject reduction (Lemma 2.11) there is an experiment \mathbf{e}' on π' with the same result. Viceversa let \mathbf{e}' be an experiment on π' , and let us reason by induction on the depth of the inner interface of $\omega[\]$. Let $\mathbf{r}(\mu) = \lambda_1 + \dots + \lambda_n$ simple nets.

- If $\omega[\] = \delta[\] + \rho$ with $\delta[\]$ simple and linear, then $\pi' = \sum_i \delta[\lambda_i] + \rho$. Then \mathbf{e}' is either on ρ , and then it is also an experiment on π , or it is on a $\delta[\lambda_i]$, in which case it induces an experiment $\mathbf{e}' \upharpoonright_{\lambda_i}$ on λ_i and therefore on $\mathbf{r}(\mu)$. This in turn by surjectivity of the subject reduction function induces an experiment on μ , which finally by compositionality of context plugging, combined with $\mathbf{e}' \upharpoonright_{\delta[\]}$ gives an experiment \mathbf{e} on $\delta[\mu]$ (so on π) with the same result.
- Otherwise, let $B[\] \in \mathbf{b}_0(\omega[\])$ be the box with the context inside, and let $\phi[\] + \rho$ be the unique monic affine context (with $\phi[\]$ simple) such that $\omega[\] = \phi[B[\]]$ + ρ . Now again if \mathbf{e}' is on ρ there is nothing else to prove. If instead \mathbf{e}' is on $\phi[B[\mathbf{r}(\mu)]]$, it induces k experiments $\mathbf{e}'_1, \dots, \mathbf{e}'_k$ on $\sigma(B)[\mathbf{r}(\mu)]$. These by induction hypothesis give experiments $\mathbf{e}_1, \dots, \mathbf{e}_k$ on $\sigma(B)[\mu]$ so that

$$[\mathbf{e}_1(\sigma(B)[\mu]), \dots, \mathbf{e}_k(\sigma(B)[\mu])] = [\mathbf{e}'_1(\sigma(B)[\mathbf{r}(\mu)], \dots, \mathbf{e}'_k(\sigma(B)[\mathbf{r}(\mu)])].$$

So we are able to define an experiment \mathbf{e} on $B[\mu]$ with the same result of $\mathbf{e}' \upharpoonright_{B[\mathbf{r}(\mu)]}$. The rest is again compositionality of context plugging, by combining \mathbf{e} with $\mathbf{e}' \upharpoonright_{\phi[\]}$ and obtaining thus an experiment on $\phi[B[\mu]]$ (so on π) with the same result.

Throwing in compatibility with \mathcal{T} does not change the above reasoning. \square

Usually relational denotational semantics are axiomatic, with Lemma 2.3 applying to experiments. Also in all our examples the dual of points is the identity, but there is no a priori reason to ask such a condition. However from now on we will implicitly have autodual universes. In particular we will not need to specify directions when assigning points to wires.

2.3.2 Webbed Semantics and Semantic Correctness

In the following we will apply the above definitions to the various fragments and extensions of linear logic presented in the previous chapter. At the same time, we will show some of the (fragments of the) main semantics of LL: *coherent spaces*, *hypercoherent spaces* and *finiteness spaces*. All these models are *webbed*. We here define the minimal notions that we need in this setting.

Let \mathcal{C} be a category with a distinguished object 1 and an associative bifunctor \wp . We call **states** of an object \mathcal{X} its 1 -valued points, i.e. the morphisms $x: 1 \rightarrow \mathcal{X}$, denoted by $x: \mathcal{X}$. We say \mathcal{C} is **webbed** if it comes with a faithful functor $|\cdot|: \mathcal{C} \rightarrow \mathbf{Rel}$ such that $|1| = \{*\}$, the set with one point, and $|\mathcal{X} \wp \mathcal{Y}| = |\mathcal{X}| \times |\mathcal{Y}|$.

On objects $\mathcal{X} \in \mathcal{C}$, the set $|\mathcal{X}|$ is called **web**. Being faithful means that we can identify a morphism $f: \mathcal{X} \rightarrow \mathcal{Y}$ with its corresponding relation $f: |\mathcal{X}| \rightarrow |\mathcal{Y}|$, so that $\mathcal{C}(\mathcal{X}, \mathcal{Y})$ is a subset of $\mathbf{Rel}(|\mathcal{X}|, |\mathcal{Y}|)$. In particular the states of an object \mathcal{X} are just the $\{*\}$ -values points of $|\mathcal{X}|$ in \mathbf{Rel} , which are the *subsets* of $|\mathcal{X}|$.

Every object \mathcal{X} of \mathcal{C} therefore naturally induces the predicate $s: \mathcal{X}$ on subsets $s \subseteq |\mathcal{X}|$, indicating that s is (the image by $|\cdot|$ of) a state of \mathcal{X} .

A \mathcal{C} -denotational semantics for a type system \mathcal{T} is a function $\llbracket \cdot \rrbracket: \mathcal{T} \rightarrow \mathcal{C}$ such that $|\cdot| \circ \llbracket \cdot \rrbracket = \llbracket |\cdot| \rrbracket$ is a relational denotational semantics for \mathcal{T} . Given a sequent Γ , its interpretation is

$$\llbracket \Gamma \rrbracket := \bigotimes_{\tau \in \Gamma} \llbracket \tau \rrbracket, \quad \text{so that} \quad \llbracket |\Gamma| \rrbracket = \prod_{\tau \in \Gamma} \llbracket |\tau| \rrbracket.$$

We will implicitly suppose that experiments are compatible with the \mathcal{C} -denotational semantics when we will be in a typed setting and will be considering a particular \mathcal{C} .

We now introduce one of the central notions of Part II, **semantic correctness**. The interpretation defined above maps all nets of type Γ to a subset of $\llbracket |\Gamma| \rrbracket$. The classical notion of semantics is recovered by results of soundness, stating that given a *proof* net π , its interpretation $\llbracket \pi \rrbracket$ is a state of $\llbracket |\Gamma| \rrbracket$.

However nets offer a more general syntax, mainly one where we are able also to make “mistakes”. It is natural to ask oneself which nets correspond to states. We will call a typed net **\mathcal{C} -correct**, or simply **semantically correct** leaving \mathcal{C} implicit, if for all interpretations $\llbracket \cdot \rrbracket$ in \mathcal{C} , we have that $\llbracket \pi \rrbracket: \llbracket |\tau(\pi)| \rrbracket$.

This opens question we can ask about such semantics.

- Soundness results state that syntactic correctness implies the semantic one: is it possible to inverse such relation?
- If the above is not true, is it possible to characterize semantic correctness with a syntactic criterion, much as for example switching acyclicity corresponds to sequentializability in MLL?
- Does semantic correctness give “good” properties to the system, in particular with respect to reduction?

The first question is very near to the one of *full completeness* (every morphism in the category has a term/proof/net interpreted into it). The subtle difference is that we ask such result already on a restriction of morphisms, by looking only at those that interpret possibly incorrect nets. In particular usually the models inspected are very far from being fully complete per se, and the typical step in

proving full completeness results such as in [AM99, BHS05] is in fact showing that the morphism (concurrent games in the work by Abramski and Mellès, dinatural transformation of double glued hypercoherent spaces in the one by Blute, Hamano and Scott) are indeed proof structures, i.e. nets of the system.

Chapter 3

Linear Proof Nets

In this chapter we will use the abstract machinery presented in the previous chapter to present the linear fragment of LL, where structural rules are completely forbidden. First we present the core fragment, multiplicative LL (MLL), then we will move on to the multiplicative additive one (MALL).

3.1 The Multiplicatives

In the previous chapter we already defined the MLL system as a running example. Let us recall it here. The MLL type system is defined by the grammar

$$\mathcal{T}_{\text{MLL}} ::= \mathcal{V} \mid \mathcal{V}^\perp \mid 1 \mid \perp \mid \mathcal{T}_{\text{MLL}} \otimes \mathcal{T}_{\text{MLL}} \mid \mathcal{T}_{\text{MLL}} \wp \mathcal{T}_{\text{MLL}},$$

where \mathcal{V} is a countable set of type variables, and types α or α^\perp with α type variable are said to be **atomic**. Duality is defined by usual De Morgan's duality, with

$$1^\perp = \perp, \quad (A \otimes B)^\perp = A^\perp \wp B^\perp.$$

Figure 3.1 shows cells together with typing rules, and reduction rules. As MLL has an axiomatic type system, types may be shown on axioms only. An MLL net is a typed net in \mathbf{N}_{MLL} .

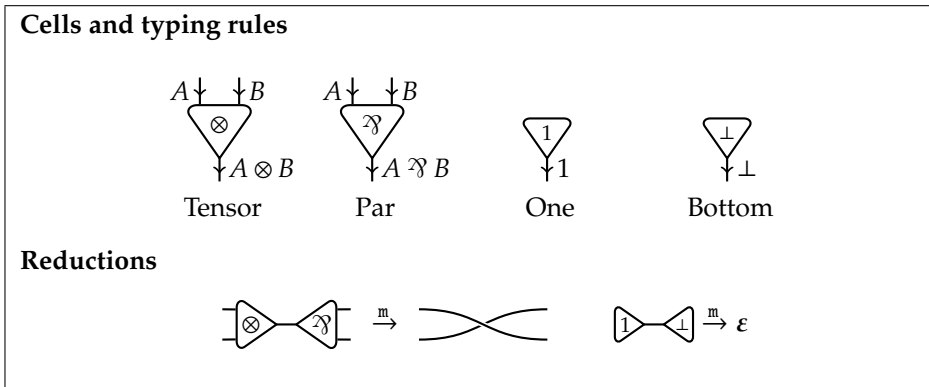


Figure 3.1: Cells, typings and reductions of MLL.

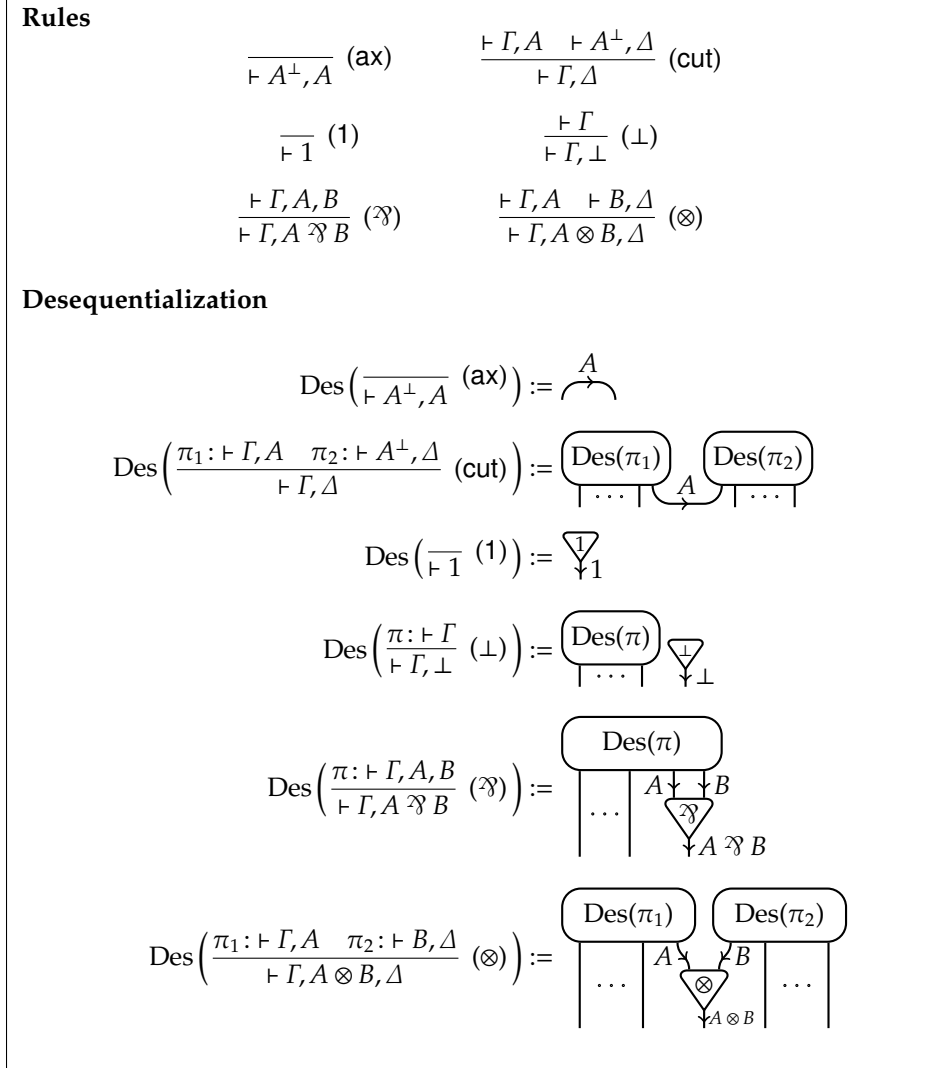


Figure 3.2: Inference and desequentialization rules for proofs of MLL.

3.1.1 Sequent Calculus and Desequentialization

We will now investigate the logical content of MLL nets.

We first introduce MLL sequent calculus on the same alphabet. As already defined in the previous chapter, a **sequent** is a multiset of types. A **proof** is then a tree with nodes labeled by $\vdash \Gamma$ where Γ is a sequent. Labels of parents and childs must follow the rules shown in Figure 3.2. A proof π of a sequent Γ is denoted by $\pi: \vdash \Gamma$. The reduction rules are easily seen to have subject redution.

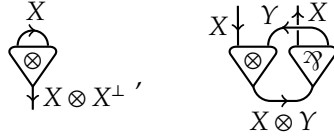
As shown in [Gir87], there are some inessential rule commutations that can be quotiented by *desequentializing* a sequent calculus proof into a typed (simple) net. We show such desequentialization function $\text{Des}(\pi)$, defined by induction, in Figure 3.2 under the inference rules. The free ports of $\text{Des}(\pi)$ are in correspondence with the type occurrences in Γ , which corresponds exactly to the

type they get. They are shown in the same order of appearance in Γ . We call **MLL proof nets**, or sequentializable MLL nets, the elements of the image of the desequentializable function.

We may note that with the interaction net paradygm of [Laf95] one quotients also with respect to sequences of axiom and cuts, so that for example

$$\text{Des} \left(\frac{\frac{\overline{\vdash A, A^\perp} \text{ (ax)}}{\vdash A, \Gamma} \text{ (cut)}}{\vdash A, \Gamma} \right) = \text{Des}(\pi : \vdash A, \Gamma).$$

It is immediately apparent that the uentialization function is not surjective, i.e. there are typed MLL nets that are not sequentializable. For example simple nets containing deadlocks, or the following ones



Not only these nets are not sequentializable, but they are also unreliable from the point of view of reduction. B  chet has shown in [B  c98] that as soon as a simple net λ is not sequentializable there is another, sequentializable, that cut against λ reduces to a net with deadlocks. So sequentialization can be seen as a kind of certificate assessing that the net can safely interact with other correct (i.e. sequentializable) simple nets.

One therefore feels the need to be able to state this correctness in an independent way with respect to sequent calculus. This need is fulfilled by *correctness criterions*, the most famous of which are the long trip one given by Girard in [Gir87], and the so called Danos-Regnier one, given by the two authors in [DR89].

3.1.2 Correctness Criterion and Sequentialization

In order to state Danos and Regnier's criterion, we need to introduce the concept of *switching graph*, and we may do it in general for all nets. Let us choose a subset $\Xi \subseteq \Sigma$ of symbols that we call *switching*. A **switching** on a simple net λ is then a function

$$S: \{c \in \mathbf{c}(\lambda) \mid \sigma(c) \in \Xi\} \rightarrow \text{bp}(\lambda)$$

such that $S(c) \in \text{pasv}(c)$. The **switching graph** $\mathcal{G}_S(\lambda)$ is the subgraph of $\mathcal{G}(\lambda)$ obtained by disconnecting all passive ports of switching cells apart from the one chosen by S , i.e. $\mathcal{G}_S(\lambda)$ has the same nodes of $\mathcal{G}(\lambda)$ and all of its edges but the ones that have at least a port $p \in \text{pasv}(c) \setminus \{S(c)\}$ for some c .

For MLL the only switching symbol is \wp .

Definition 3.1 (Danos-Regnier criterion for MLL). A net λ is said to be **strongly correct** if for all of its switchings S the graph \mathcal{G}_S is acyclic and connected.

$$\begin{array}{c}
\frac{\vdash \Gamma \quad \vdash \Delta}{\vdash \Gamma, \Delta} \text{ (mix}_2\text{)} \quad \frac{}{\vdash} \text{ (mix}_0\text{)} \\
\text{Des} \left(\frac{\pi_1 : \vdash \Gamma \quad \pi_2 : \vdash \Delta}{\vdash \Gamma, \Delta} \text{ (mix}_2\text{)} \right) := \text{Des}(\pi_1) \parallel \text{Des}(\pi_2) \\
\text{Des} \left(\frac{}{\vdash} \text{ (mix}_0\text{)} \right) := \varepsilon
\end{array}$$

Figure 3.3: The mix inference and desequentialization rules.

This in reality is not equivalent to sequentializability because of problems with the unit \perp . For example a single \perp cell is strongly correct, or on the other hand a correct net with a \perp cell at one side is not. So one has in fact the following result.

Theorem 3.2 (Danos and Regnier, [DR89]). *If λ is an MLL net with no \perp cell, then λ is sequentializable iff it is strongly correct.*

We can work around this quirk by shifting the perspective on the system, in a way that also semantics strongly supports (see Section 2.3). We add two rules that though are logically “wrong” (we have the possibility of inferring \perp , logical contradiction!), they have a very good and natural computational meaning and behaviour. These rules are the binary and zeroary mix rules, shown in Figure 3.3 with their respective desequentialization rules, that are juxtaposition and the empty net. Introducing them is equivalent to setting $1 \cong \perp$ (i.e. symmetrizing their rules). The zeroary mix rule is usually called *daimon*. In this framework we can relax the correctness criterion dropping connectedness.

Definition 3.3 (Danos-Regnier criterion for MLL+mix). A net λ is said to be **correct** if for all of its switchings S the graph \mathcal{G}_S is acyclic.

We can also easily restate this criterion in terms of paths in the net, dropping the definition of switchings. A **switching path** in $\mathcal{G}(\mu)$ is an elementary one that never bounces on a switching cell. In other words, no two passive ports of a switching cell appear in it. The above criterion becomes equivalent to **switching acyclicity**, i.e. the absence of switching paths that are cycles.

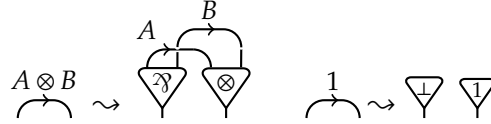
Theorem 3.4 (Danos-Regnier, [DR89]). *If λ is an MLL net, then λ is sequentializable in MLL + mix iff it is correct.*

It is important now to see that such correctness criterion is stable under cut reduction. This is a fundamental property, as it entails that the criterion does not only describe a static and inert property of the nets, but rather one that is invariant under reduction.

3.1.3 MLL nets as linkings

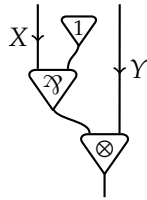
Every MLL net has an equivalent η -expanded form. Without going much into detail, such a form is obtained by substituting non η -expanded axioms induc-

tively by



It suffices us to say that restricting ourselves to η -expanded nets does not restrict the computational and logical expressiveness of the system.

Given an MLL sequent Γ , an MLL net (which we also call Γ) can be naturally recovered from it as its syntactic forest. More formally, to each formula A we inductively associate a tree by assigning wires to atomic types, and the corresponding cells to binary connectives \otimes , \wp and 0-ary connectives 1 , \perp . So for example the formula $(X \wp 1) \otimes Y$ can be represented as



Then we assign to a sequent the juxtaposition of its formulas as trees.

An **axiom linking** on Γ is a partition of the leaves of Γ (i.e. the ports corresponding to its atomic subformulas) into pairs of leaves having dual type. In fact an axiom linking is exactly a typed linking having as free ports the leaves of Γ .

If we glue a sequent Γ with an axiom linking, we obtain an η -expanded cut-free MLL net without vicious cycles. Remark 2.1 assures us that such operation is bijective: every η -expanded vicious cycle and cut free MLL net with type Γ correspond uniquely to a cut sequent $[\Sigma]\Gamma$ and an axiom linking on it. Therefore the latter provide an equivalent syntax to describe the nets in the system.

3.2 The Additives

In the previous section we addressed the multiplicative linear fragment of linear logic, we now move on with the so-called *additive* connectives. Contrary to what we have done with the multiplicatives, we will omit additive units (\top and 0), as until now no way of implementing them in nets is known¹.

In order to describe MALL in nets, we adopt the idea of *slicing* the proofs. This idea (first described by Girard in [Gir96], further developed in [LTdF04] and finally in [HvG03]) consists in seeing an additive net as a set of separate quasi-multiplicative nets. As we already introduced multisets of simple nets, we will use the multiset sum construction to group together slices. The correctness criterion will in fact reject repetition of slices, so one can equivalently use sets (as is the usual way in the literature) rather than multisets.

¹This is true in the general case. In polarized proof nets there is a correctness criterion taking into account also such units, described in [LQtdF05].

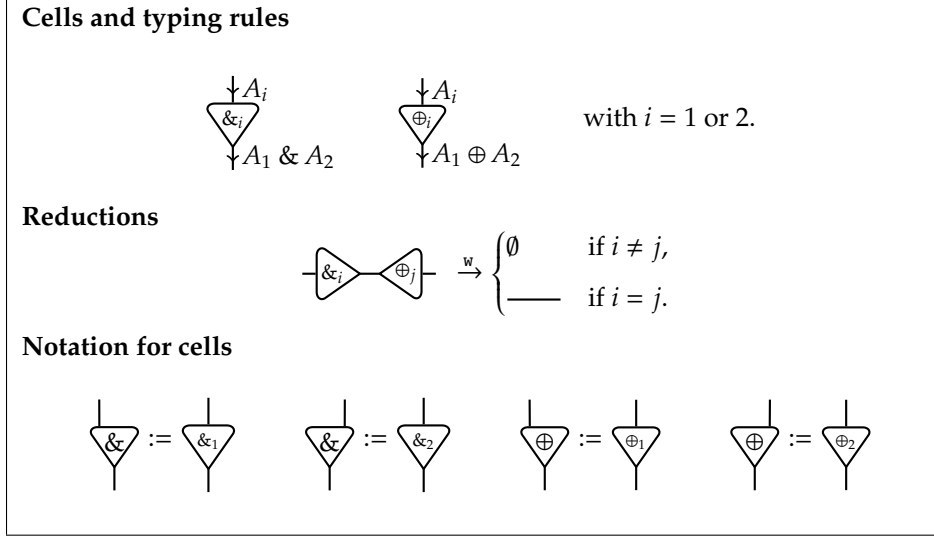


Figure 3.4: Cells, typing rules and reductions of MALL nets. We also show the alternative and intuitive notation, where the symbols $\&_1$ and $\&_2$ (resp. \oplus_1 and \oplus_2) are identified, but recovered from the position of the auxiliary port: left for $\&_1$ (resp. \oplus_1) and right for $\&_2$ (resp. \oplus_2).

The type system of MALL adds the connectives $\&$ and \oplus :

$$\mathcal{T}_{\text{MALL}} ::= \mathcal{V} \mid \mathcal{V}^\perp \mid 1 \mid \perp \mid \mathcal{T}_{\text{MALL}} \otimes \mathcal{T}_{\text{MALL}} \mid \mathcal{T}_{\text{MALL}} \wp \mathcal{T}_{\text{MALL}} \\ \mid \mathcal{T}_{\text{MALL}} \& \mathcal{T}_{\text{MALL}} \mid \mathcal{T}_{\text{MALL}} \oplus \mathcal{T}_{\text{MALL}}.$$

A MALL **net** is a slice net, built from the multiplicative symbols of Figure 3.1 together with the new symbols depicted in Figure 3.4.

3.2.1 Sequent Calculus and Desequentialization

MALL sequent calculus extends the rules of MLL (as presented in Figure 3.2) with new rules for the two additive connectives shown in Figure 3.5. Notice that by employing the desequentialization rules already defined for the multiplicatives we implicitly use the definition of glueing for slice nets. So for example in the case for tensor we are saying in fact that

$$\text{Des} \left(\frac{\pi_1 : \vdash \Gamma, A \quad \pi_2 : \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta} (\otimes) \right) := \left\{ \left(\begin{array}{c} \lambda \quad \mu \\ \dots \mid A \quad B \mid \dots \\ \vdash A \otimes B \end{array} \right) \mid \lambda \in \pi_1, \mu \in \pi_2 \right\}.$$

From the remainder of this part we will, for now, consider all sequential MALL proofs and MALL nets to be both cut-free, η -expanded and (for nets) deadlock-free. Cuts will be treated up next in Section 3.2.4.

We will be addressing the issues already confronted about MLL. So again we will call MALL **proof nets** the ones that are in the image of the desequentialization function.

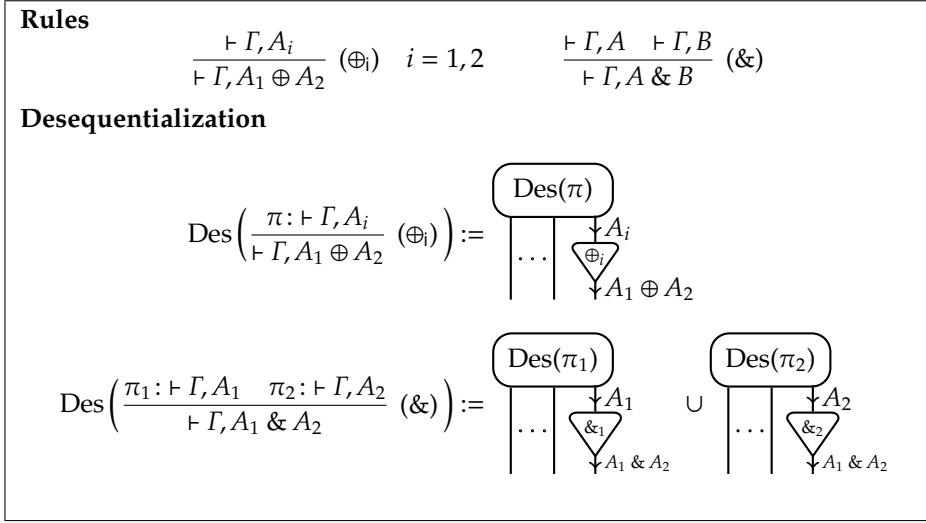


Figure 3.5: Sequent calculus rules for MALL additive connectives, and the respective desequentialization rules for nets.

As representing MALL nets by listing all of their simple nets is quite cumbersome, we first reintroduce the way they were represented in [HvG03].

3.2.2 MALL Nets as Sets of Linkings

We cannot directly apply the approach used with MLL, as MALL cells are unary, therefore they cannot completely take the role of the connectives. However once we factorize an η -expanded MALL net as shown in Remark 2.1, the forest deriving from it is clearly a subforest of the sequent Γ seen as a syntactical forest, by using the following convention

$$\begin{array}{c} \downarrow A_1 \\ \triangleleft \&_1 \\ \downarrow A_1 \& A_2 \end{array} = \begin{array}{c} \downarrow A_1 \\ \triangleleft \& \\ \downarrow A_1 \& A_2 \end{array} \subseteq \begin{array}{c} A_1 \ A_2 \\ \triangleleft \& \\ A_1 \& A_2 \end{array}$$

and similar conventions for $\&_2$, \oplus_1 and \oplus_2 . The equality on the left is the graphic convention described in Figure 3.4, while the tree on the left is the syntactical branching corresponding to the binary $\&$ connective.

Such subforests are called **additive resolutions**. More in general, we call a **resolution** G of a sequent Γ a subforest of Γ such that

- G and Γ have the same roots;
- all \otimes s and \wp s are binary, i.e. if they are in G both their children are too.

In fact a resolution can be determined by choosing to erase some children (the left, the right or both) of every additive connective $\&$ or \oplus .

It is straightforward, given a resolution G and an additive connective c in Γ and in G , to define the arity of c in G , which can be either 0, 1 or 2. We will therefore denote by

- $\&0(G)$ (resp. $\oplus0(G)$) the set of zeroary $\&$ s (resp. \oplus s);
- $\&1(G)$ (resp. $\oplus1(G)$) the unary ones, and
- $\&2(G)$ (resp. $\oplus2(G)$) the binary ones.

All the $\&$ s (resp. \oplus s) in G (that need not be all the ones of Γ) are denoted by $\&(G) := \&0(G) \cup \&1(G) \cup \&2(G)$, (resp. $\oplus(G) := \oplus0(G) \cup \oplus1(G) \cup \oplus2(G)$).

A union and the intersection of two resolutions is still a resolution, thus they form a bounded lattice, where the greatest element is Γ and the least one is the unique resolution with all zeroary additive connectives.

We say that a resolution G is **proper** if $\&0(G) = \emptyset$ and $\oplus0(G) = \emptyset$. Equivalently, G is proper if its leaves are contained in those of Γ . Finally, a proper resolution is equivalently determined by a subset S of the leaves of Γ by taking all nodes and edges of Γ under S , provided that at least a leaf is chosen above each root, and that if a leaf is chosen over one child of a \otimes or \wp , one is chosen also above the other. We say in this case that G is **generated** by S . A union of two proper resolutions is still proper, and their generating set of leaves is the union of the two.

We distinguish two types of proper resolutions that play an important role in MALL.

- The **additive** resolutions are the minimal proper ones, where all $\&$ s and \oplus s are unary, i.e. $\&1(G) = \&(G)$ and $\oplus1(G) = \oplus(G)$.
- The **$\&$ -resolutions** are the proper ones where only $\&$ s are unary, i.e. $\&1(G) = \&(G)$ and $\oplus1(G) = \emptyset$.

Additive resolutions precisely characterize the subforests that can appear in the factorization of a MALL simple net. Therefore a MALL simple net is completely determined by an axiom linking and an additive resolution. On the other hand, the additive resolution is generated by the leaves selected by the axiom linking plus a choice of type constants. By abuse of terminology, we will call **linking over Γ** a partition of *some* of the type variables of Γ into sets of two dual types, together with a subset of multiplicative constant types in Γ , such that the resulting subset of leaves of Γ generates an additive resolution $G(\lambda)$.

Summing up, each (typed, cut and deadlock-free) simple MALL net λ is equivalently defined by a linking which we call $\ell(\lambda)$, which in turn defines a unique additive resolution $G(\lambda)$ of Γ . In fact glueing $\ell(\lambda)$ with $G(\lambda)$ (identifying multiplicative constants), we obtain back the *net* λ .

So finally a (cut and deadlock-free) MALL net can be represented by just drawing the set of linkings, one above the other, over the sequent. An example is shown in Figure 3.6.

Given any MALL net $A = \{ \lambda_1, \dots, \lambda_k \}$ we have the following notations:

$$G(A) \equiv G(\lambda_1, \dots, \lambda_k) := \bigcup_{i=1}^k G(\lambda_i)$$

$$\&(A) := \&(G(A)), \quad \oplus(A) := \oplus(G(A))$$

and similar notations for $\&i$ and $\oplus i$ with $i = 0, 1, 2$.

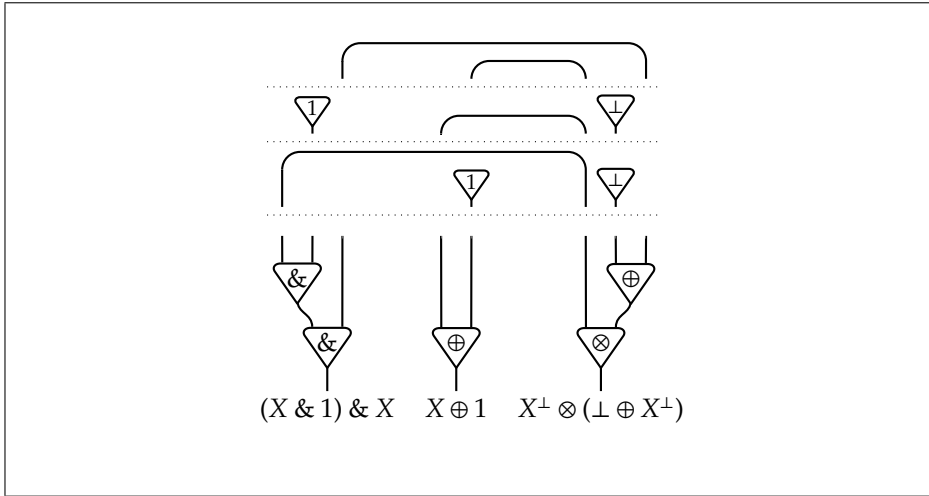


Figure 3.6: An example of MALL net, showing the compact representation.

3.2.3 Correctness Criterion and Sequentialization

Again we address here the problem of determining whether a given MALL net is sequentializable. The definitions and main results of this section come from [HvG03].

MLL and resolution conditions

The first naive idea sprouts from the fact that one can easily apply the MLL correctness criterion to MALL simple nets, as the new cells are unary. So, by still defining as switching only the symbol \wp , we can have the following definition.

Definition 3.5 (MLL condition). A MALL net θ is said to satisfy the **strong MLL condition** (resp. the MLL condition) if for every $\lambda \in \theta$ and every switching S on it, the graph \mathcal{G}_S is acyclic and connected (resp. acyclic).

It is quite immediate to see that this condition does not suffice. A first problem is that there may not be *enough* simple nets in θ to describe all the possible branchings of $\&$ s. The simplest counterexample is the empty slice net \emptyset , which trivially satisfies the strong MLL condition but is not a proofnet. Another related problem is that there may be *too much* simple nets. A straightforward example of this is two linkings on the same MLL sequent: clearly there is no way that a proof using only MLL rules can give rise to two simple nets.

This problems are addressed by the following definition.

Definition 3.6 (Resolution condition). A MALL net θ over a sequent Γ is said to satisfy the **resolution condition** if for every $\&$ -resolution H of Γ there is a unique net $\lambda \in \theta$ such that $G(\lambda) \subseteq H$. We can divide this condition in the following two:

&-compatibility: if $\lambda, \mu \in \theta$ have a $\&$ -resolution in common such that $G(\lambda), G(\mu) \subseteq H$ then $\lambda = \mu$ (uniqueness);

&-fullness: for every $\&$ -resolution H there is at least a net $\lambda \in \theta$ such that $G(\lambda) \subseteq H$.

The following is a lemma easily restating the $\&$ -compatibility condition.

Lemma 3.7. *A MALL net θ is $\&$ -compatible if and only if*

$$\forall \lambda, \mu \in \theta, \lambda \neq \mu : \&2(\lambda, \mu) \neq \emptyset.$$

Proof.

\Rightarrow) Suppose two different $\lambda, \mu \in \theta$ have $\&2(\lambda, \mu) = \emptyset$. This means that

$$\&1(G(\lambda) \cup G(\mu)) = \&(G(\lambda), G(\mu))$$

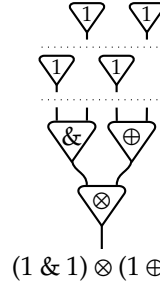
which easily implies the existence of a $\&$ -resolution H containing $G(\lambda) \cup G(\mu)$, as it suffices to add missing branches to \oplus s in $\oplus 1(\lambda, \mu)$, possibly choosing arbitrarily which branch to keep of $\&$ s above them. Thus $G(\lambda) \subseteq H$ and $G(\mu) \subseteq H$ which contradicts $\&$ -compatibility.

\Leftarrow) Given any $\&$ -resolution H with $G(\lambda) \subseteq H$ and $G(\mu) \subseteq H$ then

$$G(\lambda, \mu) = G(\lambda) \cup G(\mu) \subseteq H \implies \&2(\lambda, \mu) \subseteq \&2(H) = \emptyset$$

which implies that λ and μ cannot be different. \square

But again, a net satisfying both the MLL condition and the resolution one need not be a proof net. Take for example the following MALL net.



$$(1 \& 1) \otimes (1 \oplus 1) \tag{3.1}$$

Both simple nets in it are strongly MLL correct, and they correspond exactly to the two $\&$ -resolutions of $(1 \& 1) \otimes (1 \oplus 1)$. However if one tries to sequentialize, one find it impossible to separate the premises of the \otimes , which must be the last applied rule. The impossibility to separate the contexts of a tensor (or a cut) is what is captured by switching cycles. Here one must be able to capture another type of cyclic dependency.

Jumps and the toggling condition

The tool usually employed to expose dependencies not explicitly present in the graph of the proof is the notion of *jump*. First appeared to implement second order quantifiers in proof nets in [Gir91b], then for linear logic additives in [Gir96], we here present how they are used in [HvG03]. Jumps will play a central role in the results of Part II. Recently jumps have been thoroughly studied as a tool to get different degrees of sequentiality, for example in [DG08].

Given a subset $A \subseteq \theta$ we will build the **correctness graph** $\mathcal{G}^{\text{HvG}}(A)$ by, informally speaking, adding special jump edges to $\bigcup A$, the superposition of the

nets in Λ . HvG stands for Hughes and van Glabbeek, as this is their version of correctness graph (we will use a different one in the future).

Formally, $\mathcal{G}^{\text{HvG}}(\Lambda)$ is defined extending the forest $\bigcup_{\lambda \in \Lambda} G(\lambda)$ with the following new edges:

axioms: an edge between two leaves a, a' for every axiom $\{a, a'\}$ in $\bigcup_{\lambda \in \Lambda} \ell(\lambda)$; notice axioms connecting the same leaves in different linkings are identified;

jumps: an edge between a leaf a and $w \in \&2(\Lambda)$ if there are $\lambda, \mu \in \Lambda$

$$a \in e \in \ell(\lambda) \setminus \ell(\mu) \quad \text{and} \quad \&2(\lambda, \mu) = \{w\},$$

where e is an axiom if a is atomic, and the singleton $\{a\}$ if a is a type constant.

Jumps register a direct dependency of the existence of certain axioms from the choice made on a given $\&$.

In the upcoming definitions we will adapt to such correctness graphs (which are *not* interaction nets) the definitions already employed in MLL nets. More precisely note that there are nodes (the leaves of Γ) which do not correspond to actual cells of any λ .

In $\mathcal{G}^{\text{HvG}}(\Lambda)$ we may define elementary and switching paths as we did for the graphs corresponding to usual interaction nets. For this purpose, jumps are considered to connect a passive port of the corresponding $\&$, which is considered switching when binary. We are just saying that switching paths traversing a jump or a premise of a $\&$ cannot have another jump or a premise of the same $\&$, and that the leaves and binary \oplus s which are not switching nodes.

We are finally ready to define the last (and hardest) part of the correctness criterion.

Definition 3.8 (toggling condition). A MALL net θ is said to satisfy the **toggling condition** if and only if

$$\forall \Lambda \subseteq \theta, \# \Lambda \geq 2 : \exists w \in \&2(\Lambda) \mid \forall \phi \text{ switching cycle in } \mathcal{G}^{\text{HvG}}(\Lambda) : w \notin \phi.$$

In other words: for every choice of MALL nets in θ , there is a binary $\&$ which lies outside all switching cycles in $\mathcal{G}^{\text{HvG}}(\Lambda)$.

The following are the results exposed in [HvG03], easily adapted to cover multiplicative units in the same way they are in the multiplicative case.

Definition 3.9 (Hughes-van Glabbeek correctness for MALL). We say that an MALL net θ is strongly correct (resp. correct) if θ satisfies the resolution, strong MLL and toggling conditions (resp. resolution, MLL and toggling).

Proposition 3.10. *A MALL net is correct if and only if*

- *it satisfies the resolution condition;*
- *for every $\Lambda \subseteq \theta$ and every non empty union S of switching cycles in $\mathcal{G}^{\text{HvG}}(\Lambda)$ there is $w \in \&2(\Lambda)$ with $w \notin S$.*

Proof. For $\lambda \in \theta$ we have that $\mathcal{G}^{\text{HvG}}(\lambda)$ is $\mathcal{G}(\lambda)$ up to contracting all nodes that are leaves. As $\&2(\lambda) = \emptyset$ the second point above restricted to λ singleton is equivalent to the MLL condition, as it disallows non empty union of cycles. If we then restrict to λ non singleton we practically get exactly the toggling condition. \square

We still need to control the MLL condition separately if we want strong correctness.

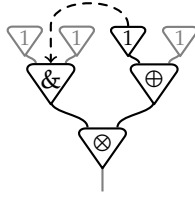
Theorem 3.11 (MALL sequentialization, [HvG03]). *If θ is an MALL cut free net with no \perp cell (resp. an arbitrary MALL net), then it is sequentializable in MALL (resp. in MALL+mix) if and only if it is strongly correct (resp. correct).*

Proof. We sketch how to account for multiplicative units in the proofs of [HvG03]. As already stated there, the unit 1 is easily implementable by translating the formula with $X \wp X^\perp$, and the corresponding cell with an axiom between the two literals. One easily shows that both

$$\begin{aligned} \theta \text{ is strongly correct} &\iff \theta \left[\frac{\text{1}}{\wp} / \frac{\text{1}}{\perp} \right] \text{ is strongly correct} \\ \text{and } \theta \text{ is sequentializable in MALL} &\iff \theta \left[\frac{\text{1}}{\wp} / \frac{\text{1}}{\perp} \right] \text{ is also.} \end{aligned}$$

This covers the case for MALL nets with no \perp cell. For the second part it should be noted that the mix_0 rule can only be followed by a \perp rule, the only one possible with an empty sequent. Thus, as expected, \perp plays exactly the same role of 1 in MALL+mix, and we can employ exactly the same translation for \perp , getting the same equivalencies as before for plain correctness. \square

For example let θ be the MALL net in (3.1). Then we have the following cycle in $\mathcal{G}^{\text{HvG}}(\theta)$ that covers the unique binary $\&$ in θ .



We have drawn only one of the four jumps of $\mathcal{G}^{\text{HvG}}(\lambda)$. Apart from it there is the one from the other premise of \oplus , and the two “redundant” ones from the premises of $\&$. A wealth of other examples are shown in [HvG03], and we will have other ones in Section 2.3 after we have presented denotational semantics.

Saturated MALL nets

We present in this section a tool which was extensively used in the proof of the sequentialization theorem in [HvG03] and that is also useful in the results of Part II.

A subset of $\lambda \subseteq \theta$ of a MALL net θ is said to be **saturated** (in θ) if

$$\forall \lambda \in \theta \setminus \lambda : \&2(\lambda, \lambda) \neq \&2(\lambda),$$

i.e. we cannot add to Λ any net in θ without increasing binary $\&$ s.

If θ satisfies the resolution condition, we can give a characterization of its saturated subsets without mentioning θ itself. We call Λ a **saturated MALL net** if one that satisfies the following condition, which is a weakened form of the resolution one.

$$\forall H \text{ \&-resolution s.t. } \&2(H, G(\Lambda)) = \&2(\Lambda) : \exists! \lambda \in \Lambda \mid G(\lambda) \subseteq H.$$

We are in fact restricting the resolution condition to $\&$ -resolutions that in a way are already known to not disagree with any net in Λ .

Lemma 3.12. *If θ is a net satisfying the resolution condition, then $\Lambda \subseteq \theta$ is a saturated set if and only if it is a saturated MALL net.*

Proof.

\implies) $\&$ -compatibility is clearly inherited by Λ from θ , so uniqueness is trivial. For existence, take H such that $\&2(H, G(\Lambda)) = \&2(\Lambda)$. By $\&$ -fullness there is λ with $G(\lambda) \subseteq H$. We show that $\lambda \in \Lambda$. If not, as

$$\&2(\Lambda) \subseteq \&2(\lambda, \Lambda) \subseteq \&2(H, \Lambda) = \&2(\Lambda),$$

we get a contradiction to Λ being a saturated subset.

\impliedby) Suppose there is $\lambda \in \theta \setminus \Lambda$ such that $\&2(\lambda, \Lambda) = \&2(\Lambda)$. Take any $\&$ -resolution H that

- on $\&$ s in $\&(\lambda)$ chooses as $G(\lambda)$,
- on $\&$ s in $\&1(\Lambda)$ chooses as $G(\Lambda)$,
- on all other $\&$ s chooses randomly.

The first two clauses do not conflict as if $w \in \&(\lambda) \cap \&1(\Lambda)$ and λ and $G(\Lambda)$ choose differently on it, then $w \in \&2(\lambda, \Lambda) = \&2(\Lambda)$ which is a contradiction. We have then that both $G(\lambda) \subseteq H$ and $\&2(H, G(\Lambda)) = \&2(\Lambda)$. By saturation of Λ we get $\mu \in \Lambda$ (which implies $\lambda \neq \mu$) where both $G(\lambda)$ and $G(\mu)$ are in H , which contradicts $\&$ -compatibility. \square

A construction typically used in the proofs is the following. Take Λ saturated and $w \in \&2(\Lambda)$, and define

$$\Lambda^w := \{ \lambda \in \Lambda \mid \text{the right premise of } w \text{ is in } G(\lambda) \}.$$

So one keeps only the nets that either choose the left side of w , or do not choose at all. Furtherly, given two MALL simple nets λ and μ , define $\lambda \stackrel{w}{=} \mu$ iff $\&2(\lambda, \mu) \subseteq \{w\}$, i.e. if either they are equal or w is the only $\&$ binary in their superposition. Clearly $\&2(\Lambda^w) \subsetneq \&2(\Lambda)$, as $w \in \&1(\Lambda)$: this lets us use Λ^w as a tool to reason by induction on $\#\&2(\Lambda)$. We have the following three properties [HvG03] (which we show independently of having a net θ with resolution containing Λ).

Proposition 3.13. *If Λ is saturated then*

(S1) Λ^w is saturated;

(S2) for every $\lambda \in \Lambda$ there exists a $\lambda_w \in \Lambda^w$ with $\lambda \stackrel{w}{\sim} \lambda_w$;

(S3) for every $\lambda, \mu \in \Lambda$, if $\lambda \stackrel{x}{\sim} \mu$ then $\lambda_w \stackrel{x}{\sim} \mu_w$ for some $\lambda_w, \mu_w \in \Lambda^w$ with $\lambda_w \stackrel{w}{\sim} \lambda$ and $\mu_w \stackrel{w}{\sim} \mu$.

Proof.

(S1) Take H such that $\&2(H, G(\Lambda^w)) = \&2(\Lambda^w)$. There are two cases: either $\&2(H, G(\Lambda)) = \&2(\Lambda)$ or not. In the first case, there is a unique $\lambda \in \Lambda$ with $G(\lambda) \subseteq H$. If $\lambda \notin \Lambda^w$ then λ (and therefore H too) chooses the right premise of w , so that $w \in \&2(H, G(\Lambda^w))$ but $w \notin \&2(\Lambda^w)$ which is a contradiction.

In the other case, take a $\&$ -resolution H' which chooses

- as H in all $\&$ s in $\&(H) \cap \&2(\Lambda)$,
- as $G(\Lambda)$ in $\&1(\Lambda)$,
- left on w ,
- arbitrarily in all other $\&$ s.

The third clause can contradict neither the first, as if $w \in \&(H)$ then H chooses left, lest $w \in \&2(H, G(\Lambda^w))$, nor the second as $w \in \&2(\Lambda)$.

Clearly $\&2(H', G(\Lambda)) = \&2(\Lambda)$, so by saturation there is a $\lambda \in \Lambda$ with $G(\lambda) \subseteq H'$, which by construction implies $\lambda \in \Lambda^w$. Suppose $G(\lambda) \not\subseteq H$, that is $\exists w' \in \&2(H, G(\lambda))$. Now, $w' \in \&2(H, G(\Lambda^w)) = \&2(\Lambda^w) \subseteq \&2(\Lambda)$, and therefore by the first clause H' (and so λ) must agree with H on w' , which is a contradiction.

Similarly for uniqueness, if $\lambda \in \Lambda^w$ then one has $G(\lambda) \subseteq H$, then also $G(\lambda) \subseteq H'$. If $\exists w'' \in \&2(H', G(\lambda)) \subseteq \&2(H', H)$ then $w'' \notin \&2(\Lambda)$ or the first clause would be contradicted. But then, as $w'' \in \&(\lambda)$, we have $w'' \in \&1(\Lambda)$ and therefore H' must choose as Λ , i.e. as λ , which is a contradiction.

(S3) Take H' and H'' $\&$ -resolutions built so that

- on $\&1(\lambda, \mu) \setminus \{w, x\} = \&(\lambda) \cup \&(\mu) \setminus \{w, x\}$ both choose as $G(\lambda) \cup G(\mu)$;
- on w they both choose left;
- on x H' chooses as λ and H'' as μ ;
- on $\&1(\Lambda)$ they both choose as $G(\Lambda)$,
- on other $\&$ s they both choose the same, arbitrarily.

Then by saturation we get λ_w and μ_w such that:

$$\&2(\lambda, \lambda_w) \subseteq \&2(G(\lambda), H') \subseteq \{w\}$$

and similarly for μ and μ_w , while

$$\&2(\lambda_w, \mu_w) \subseteq \&2(H', H'') \subseteq \{x\}.$$

(S2) This is a particular case of (S3).

□

$$\begin{array}{c}
\frac{}{\vdash [] A^\perp, A} \text{ (ax)} \\
\frac{}{\vdash [] 1} \text{ (1)} \\
\frac{\vdash [\Delta] \Gamma, A, B}{\vdash [\Delta] \Gamma, A \wp B} \text{ (\wp)} \\
\frac{\vdash [\Delta] \Gamma, A_i}{\vdash [\Delta] \Gamma, A_1 \oplus A_2} \text{ (\oplus}_i) \quad i = 1, 2 \\
\frac{\vdash [\Sigma] \Gamma \quad \vdash [\Pi] \Delta}{\vdash [\Sigma, \Pi] \Gamma, \Delta} \text{ (mix}_2)
\end{array}
\qquad
\begin{array}{c}
\frac{\vdash [\Sigma] \Gamma, A \quad \vdash [\Pi] A^\perp, \Delta}{\vdash [\Sigma, A * A^\perp, \Pi] \Gamma, \Delta} \text{ (cut)} \\
\frac{\vdash [\Delta] \Gamma}{\vdash [\Delta] \Gamma, \perp} \text{ (\perp)} \\
\frac{\vdash [\Sigma] \Gamma, A \quad \vdash [\Pi] B, \Delta}{\vdash [\Sigma, \Pi] \Gamma, A \otimes B, \Delta} \text{ (\otimes)} \\
\frac{\vdash [\Delta, \Sigma_1] \Gamma, A \quad \vdash [\Delta, \Sigma_2] \Gamma, B}{\vdash [\Delta, \Sigma_1, \Sigma_2] \Gamma, A \& B} \text{ (\&)} \\
\frac{}{\vdash []} \text{ (mix}_0)
\end{array}$$

Figure 3.7: Sequent calculus rules for MALL_{cut} .

3.2.4 Cut reduction

Though the nets we have presented have a cut reduction perfectly defined, it does not behave well with respect to the logical contents of such nets, as it is completely independent in each simple net, so that sequentialization is easily lost. We will here present the definitions given in [HvG03], skipping over most of the details. Though these definitions could be carried out on nets in general, it is simpler and best to view them in this specific setting.

A **cut formula** is a commutative pair of dual formulas, denoted by $A * A^\perp$. A **cut sequent** is a regular sequent together with a multiset of cut formulas. We denote a cut sequent by enclosing the cut formulas in square brackets, as in $[\Delta] \Gamma$ where $\Delta = A_1 * A_1^\perp, \dots, A_k * A_k^\perp$. In order to desequentialize MALL proofs, an intermediate sequent calculus is provided, MALL_{cut} , whose rules are presented in Figure 3.7. This calculus simply “remembers” cuts, and on the $\&$ rules they can be either identified or not. Every proof of MALL_{cut} is easily associated with a proof of MALL by deleting all information about cuts. This function is surjective but not injective, due to the different choices that can be done when identifying cuts in the $\&$ rule.

MALL nets with cuts. A cut sequent is again a forest. A resolution G of a cut sequent $[\Delta] \Gamma$ is a subforest of $\Delta || \Gamma$ (the juxtaposition of the two syntactic forests) such that

- the roots of G are all those of Γ plus some of Δ ;
- all \otimes s, \wp s and cut formula roots are binary.

All the definition we made on regular sequents adapt seamlessly to this setting, by using these resolutions. An additive resolution is still one where all withs and pluses are unary, while $\&$ -resolutions are those that contain all cut formula roots, and all \oplus s are binary and $\&$ s unary.

Then a MALL simple net λ on a cut sequent $[\Delta] \Gamma$ will then be given by a linking $\ell(\lambda)$, i.e. a partition of the leaves of an additive resolution $G(\lambda)$ into

pairs of dual leaves (axioms) for all variable types, and singletons of the rest (constants).

A MALL net θ is a set of simple nets on the same cut sequent. At any time we delete a cut formula from $[\Delta] \Gamma$ if it does not appear in any $G(\lambda)$ for $\lambda \in \theta$.

Defining desequentialization on MALL_{cut} is straightforward. For the details the reader is referred to [HvG03]. Such sequentialization gives a relation between MALL sequent calculus proofs and MALL nets with cuts, which is not a function. However it still makes sense to characterize the image of such desequentialization relation. The answer is given again exactly by the conditions stated in the previous pages.

Theorem 3.14 (MALL_{cut} sequentialization, [HvG03]). *If θ is an MALL net on a cut sequent $[\Delta] \Gamma$ with no \perp cell (resp. an arbitrary MALL net), then it is sequentializable in MALL_{cut} (resp. in MALL+mix) if and only if it is strongly correct (resp. correct).*

We are more interested in actual cut reduction. This is defined by the following procedure. We select a cut formula C in the cut sequent, and depending on its type we perform the following procedure.

- $C = A \otimes B * A^\perp \wp B^\perp$: substitute C by $A * A^\perp, B * B^\perp$, keeping in θ the same simple nets. This is defined, as the leaves of C are all in $A * A^\perp, B * B^\perp$, and if the resolution of λ is an additive one on $[C, \Delta] \Gamma$, then so is on $[A * A^\perp, B * B^\perp, \Delta] \Gamma$ as $G(\lambda)$ must be binary on \otimes s and \wp ;
- $C = A \oplus B * A^\perp \& B^\perp$: again substitute C by $A * A^\perp, B * B^\perp$, and consider the linkings of θ on the new cut sequent. If such a linking λ is not on a resolution anymore, then delete the whole linking. This happens only if λ chose A on \oplus and B^\perp on $\&$, or B and A^\perp respectively.
- $C = 1 * \perp$: remove C , and any reference to these constants from any linking containing them.
- $C = \alpha * \alpha^\perp$: let a and a' be the two leaves. For each $\lambda \in \theta$ that selects C , define λ' by substituting in λ the two axioms $\{\lambda(a), a\}, \{a', \lambda(a')\}$ with $\{\lambda(a), \lambda(a')\}$. Then substitute each λ with λ' (and remove C by garbage collection).

Let us call this procedure **shared cut reduction**. We will still denote the reduction by $\xrightarrow{\text{sc}}$.

This cut reduction can be carried out using the interaction net rules given in Figure 3.1 and 3.4, however we have two notable differences.

- We here have back the axiom cut reduction which was abstracted away by interaction nets. Our main reason for letting it stay is that without it some definitions and results we present in Chapter 5 get needlessly more complicated.
- The reduction is called *shared* because it reduces cuts in more than one slice at a time. Fundamentally this kind of reduction is a sort of synchrony constraint, telling that a certain cut in a certain simple net can be reduced only if we do it in all nets where such cut is present.

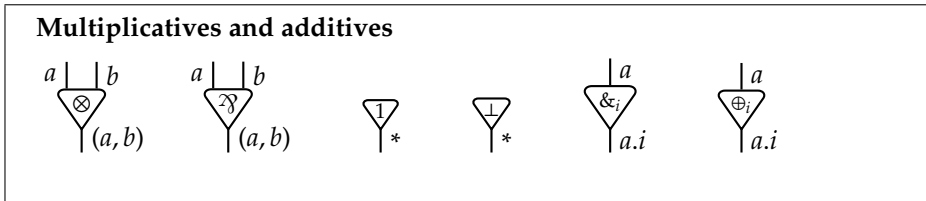


Figure 3.8: Rules for the experiments for all the cells of MALL.

It must be noted however that the normal forms obtained by following the two paradigms of reduction are the same.

We finally cite the non trivial result about stability under reduction.

Proposition 3.15 (Stability of correctness). *If θ is a (strongly) correct MALL net with cuts and $\theta \xrightarrow{\text{mrr}} \theta'$ then θ' is also.*

3.3 Invariants and Semantic Correctness

We will here define the semantics of MLL and MALL, and investigate what we indicated by semantic correctness.

3.3.1 The Interpretation

Let \mathcal{U} be a class of points containing a distinguished point $*$ and closed with respect to finite tuples (x_1, \dots, x_n) and finite disjoint union injections $x.i$. The axiomatic rules interpreting untyped MALL with points of \mathcal{U} are shown in Figure 3.8. Notice that as the system is axiomatic, we only need to specify experiments on axioms (if vicious cycles are excluded, which is automatically the case for multiplicatives and additives by typing), provided that equality on cuts is satisfied.

It is straightforward to see that the one defined above is indeed a denotational semantics, by checking each pair redex-reduct in Figures 3.1 and 3.4. As shared cut reduction is given by reducing multiple simple nets in the usual sense (apart from the axiom cut rule which however does not pose any problem), we have that $\llbracket \cdot \rrbracket$ is an invariant also under shared cut reduction.

3.3.2 Coherent Spaces

Since the beginning there was a tight pairing between linear logic and the semantic model that brought the intuitions necessary for its discovery: coherent spaces. They first where introduced in [Gir87], later refined with the multiset based exponential in [Gir91a]. We present them here and state their relation with the systems introduced in the previous chapters.

The Spaces

The category **Coh** of coherent spaces has as objects unoriented graphs. Therefore a **coherent space** \mathcal{X} is given by

- its web $|\mathcal{X}|$, a set;

- a reflexive and symmetric relation $\circ_{\mathcal{X}}$, called **coherence**.

The statement $x \circ_{\mathcal{X}} y$ is usually denoted by $x \circ y (\mathcal{X})$, or simply $x \circ y$ if no confusion is possible. The states of \mathcal{X} are its *cliques*, i.e. the sets $s \subseteq |\mathcal{X}|$ such that for every $x, y \in s : x \circ y (\mathcal{X})$.

The coherence defines, and is in turn equivalently recoverable from, the following relations:

- **strict coherence** $\wedge := \circ \setminus =$, i.e. $x \wedge y (\mathcal{X})$ iff $x \circ y (\mathcal{X})$ and $x \neq y$;
- **incoherence** $\asymp := |\mathcal{X}|^2 \setminus \wedge$, i.e. $x \asymp y (\mathcal{X})$ iff not $x \wedge y (\mathcal{X})$;
- **strict incoherence** $\smile := |\mathcal{X}|^2 \setminus \circ$, i.e. $x \smile y (\mathcal{X})$ iff $x \asymp y (\mathcal{X})$ and $x \neq y$.

The objects of **Coh** are given the structure of a *-autonomous linear category [Bie94] by the following constructs.

Dual: $|\mathcal{X}^\perp| := |\mathcal{X}|$, and $x \circ y (\mathcal{X}^\perp)$ iff $x \asymp y (\mathcal{X})$.

Multiplicatives: $|\mathcal{X} \otimes \mathcal{Y}| = |\mathcal{X} \wp \mathcal{Y}| := |\mathcal{X}| \times |\mathcal{Y}|$, and

$$\begin{aligned} (x, y) \circ (x', y') (\mathcal{X} \otimes \mathcal{Y}) &\iff x \circ x' (\mathcal{X}) \text{ and } y \circ y' (\mathcal{Y}), \\ (x, y) \circ (x', y') (\mathcal{X} \wp \mathcal{Y}) &\iff x \wedge x' (\mathcal{X}) \text{ or } y \wedge y' (\mathcal{Y}). \end{aligned}$$

The units enjoy $1 = \perp$, with $|1| = |\perp| := \{*\}$.

Additives: $|\mathcal{X}_0 \oplus \mathcal{X}_1| = |\mathcal{X}_0 \& \mathcal{X}_1| := |\mathcal{X}_0| + |\mathcal{X}_1|$, the disjoint sum. We denote an element of such a disjoint sum as $x.i$, with $i = 0$ or $i = 1$ and $x \in |\mathcal{X}_i|$.

$$\begin{aligned} x.i \circ y.j (\mathcal{X}_0 \oplus \mathcal{X}_1) &\iff i = j \text{ and } x \circ y (\mathcal{X}_i), \\ x.i \circ y.j (\mathcal{X}_0 \& \mathcal{X}_1) &\iff i = j \text{ implies } x \circ y (\mathcal{X}_i), \end{aligned}$$

so that we have always $x.1 \smile y.2 (\mathcal{X}_0 \oplus \mathcal{X}_1)$ and $x.1 \wedge y.2 (\mathcal{X}_0 \& \mathcal{X}_1)$. Again the units are identical, with $|\top| = |\mathbf{0}| := \emptyset$.

Exponentials: $!|\mathcal{X}| := \{u \in \mathcal{M}_{\text{fin}}(|\mathcal{X}|) \mid |u| : \mathcal{X}\}$, i.e. the final multisets whose support is a state. And

$$u \circ v (!\mathcal{X}) \iff u + v \in !|\mathcal{X}| \iff \forall x \in u, y \in v : x \circ y (\mathcal{X}).$$

One then defines $?|\mathcal{X}| := (!|\mathcal{X}^\perp|)^\perp$. In fact

$$u \wedge v (?|\mathcal{X}|) \iff \exists x \in u, \exists y \in v \mid x \wedge y (\mathcal{X}).$$

Notice that the web of $?|\mathcal{X}|$ is the multisets having as support cliques of \mathcal{X}^\perp , not \mathcal{X} .

The operations defined above respect De Morgan's duality.

The morphisms are the states of $\mathcal{X} \multimap \mathcal{Y} := \mathcal{X}^\perp \wp \mathcal{Y}$, which are relations $|\mathcal{X}| \multimap |\mathcal{Y}|$, so that **Coh** is indeed a webbed category, and the notions of state coincide.

Interpretation and Semantic Correctness

A **Coh**-interpretation $\llbracket \cdot \rrbracket$ on LL types is given by defining it on type variables \mathcal{V} and then extending it to \mathcal{T} by inductively applying the constructors described above to all corresponding connectives. We then combine it with the rules shown in Figure 3.8 as explained in Section 2.3.2. Notice the following.

- If we restrict to MLL or MALL as we are doing here there is no further restriction for compatibility, other than checking it on axioms. We mean that if for every axiom $e = \{p, q\}$ we assign $\epsilon(e) \in \llbracket \tau((p, q)) \rrbracket = \llbracket \tau((q, p)) \rrbracket$ such that for every cut $\{p', q'\}$ we have $\epsilon(p') = \epsilon(q')$, then we automatically have $\epsilon(r) \in \llbracket \tau(r) \rrbracket$ for every port r .
- The above is not true anymore when extending to exponentials, where the operations on webs are not independent of the coherence of the space. This is an important point: it is said that **Coh** is *uniform* for this reason. One of the main consequences of this is the loss of injectivity of the interpretation².

We will not show it here, but an important point is that if we define a denotational semantics (for **Coh** or other models) on the various sequent calculi in the classic compositional way, the desequentialization preserves the interpretation. This tells us among other things that the quotient expressed by the nets is supported by the semantics, as sequential proofs with the same desequentialization must have the same interpretation.

From now on when considering the coherence of experiments on a port p , if (and only if) we omit the space we implicitly intend $\llbracket \tau(p) \rrbracket$ for it.

We have the following soundness result.

Theorem 3.16 (Girard, [Gir87]). *Every MLL proof net is Coh-correct.*

Notice therefore that **Coh** (and in fact all models we consider here) validates the mix rules. From now on we will consider both MLL and MALL as having the mix rules. We give here proof in our setting, as though it is an easy one the concepts we will use in it will return in the proof of Theorem 5.3 in Chapter 5.

Proof. Take two experiments ϵ and f , we show that $\epsilon(\pi) \circledast f(\pi)$.

Based on such experiments, we build a directed graph \mathcal{H} out of $\mathcal{G}(\pi)$, following these steps:

1. we erase all wires d such that $\epsilon(d) = f(d)$, and all “orphaned” nodes thereafter, i.e. those not connected to any wire;
2. for every \mathfrak{A} or contraction c such that for the two passive ports p and q we have $\epsilon(p) \frown \epsilon(p)$ and $\epsilon(q) \smile \epsilon(q)$ we turn q into a new node disconnected from c ;
3. finally, we direct every remaining wire d so that $\epsilon(t(d)) \frown f(t(d))$.

²It is conjectured but not known yet if non-uniform models, such as **Rel**, **nuCoh** (non uniform coherent spaces), **Fin**, have such property. See [Tdf00, Pag06b] for more details on this issue.

First thing, \mathcal{H} is a dag, a directed acyclic graph, as every directed path in it is a switching one in $\mathcal{G}(\pi)$: bounces on switching cells are explicitly forbidden by step 2. Because of it, as soon as \mathcal{H} is non empty, which happens in particular if $\epsilon(\pi) \neq f(\pi)$, \mathcal{H} must have at least a sink, i.e. a node with only incoming edges. If we show that only a free port of π can be a sink then we conclude, as such port p would have an incoming wire by point 1, and by point 3 we would have $\epsilon(p) \sim f(p)$.

Take then any node c and let us see by cases that it cannot be a sink. Notice that all units 1 and \perp and all weakenings are necessarily erased by step 1, and that the new nodes introduced in step 2 cannot be sinks by construction.

c **tensor**. Suppose $\epsilon(c_0) \asymp f(c_0)$ ($\mathcal{X} \otimes \mathcal{Y}$) (otherwise we trivially get a way out) then if $\epsilon(c_0) = f(c_0)$ then c is erased by step 1, therefore there must be a strict incoherence on one of the premises, which concludes this case. $\epsilon(c_{[i]}) \sim f(c_{[i]})$ for i either 1 or 2.

c **par**. the only other case for now. If $\epsilon(c_0) \asymp f(c_0)$ ($\mathcal{X} \wp \mathcal{Y}$) then $\epsilon(c_{[i]}) \asymp f(c_{[i]})$ for i both 1 and 2, so neither of the two is detached by step 2. Also they cannot have both equality, otherwise c is erased by steps 1.

□

What the above proof does can be basically seen as choosing a wire where the two experiments disagree and following from it the coherence, necessarily ending on a conclusion.

We also have the following result for MALL.

Proposition 3.17. *Every MALL proof net is **Coh**-correct.*

Proof. Indirectly **Coh** is a model of MALL sequent calculus, therefore it must be such for MALL proof nets also. Alternatively, this is a corollary of Theorem 3.19 together with Proposition 3.20. □

In MLL we also have the inverse result, therefore answering positively to the first question raised at the end of Section 2.3.2, so that in fact **Coh**-correctness and sequentializability correctness coincide.

Theorem 3.18 (Retoré, [Ret97]). *Every cut-free **Coh**-correct MLL net is a proof net.*

The absence of cuts is necessary as they may hide deviant configurations from the results collected at the conclusions. Anyway it is typical of denotational semantics to not be able to express such properties of non normal elements, as the invariance is expressedly designed to rather describe the resulting normal forms.

The inverse assertion fails instead for both MALL and, on a sidenote, for MELL, as explained up next.

The Gustave MALL Net

For the latter there is a counterexample derived from Berry's so called *Gustave* function [Ber78]. This is a stable function in the Scott-continuous hierarchy

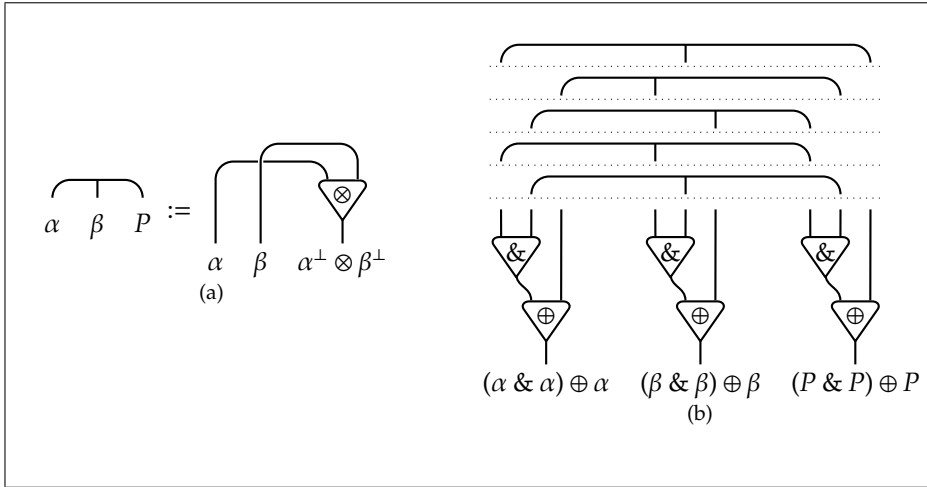


Figure 3.9: The Gustave MALL net γ is shown in Figure (b). P stands for $\alpha^\perp \otimes \beta^\perp$, and the link with three wires is shorthand for the trivial linking shown in Figure (a).

that is not sequentializable. It is defined such that for every x :

$$\begin{aligned} G(\mathbf{t}, \mathbf{f}, x) &= \mathbf{t}, \\ G(x, \mathbf{t}, \mathbf{f}) &= \mathbf{t}, \\ G(\mathbf{f}, x, \mathbf{t}) &= \mathbf{t}, \end{aligned}$$

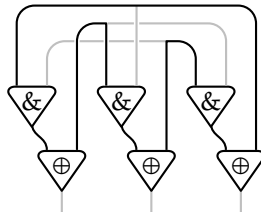
end is undefined otherwise. The fact that such function is not sequentially implementable comes from the fact that there is no first variable to look at. If for example one gives an implementation inspecting the first one, then if the input is (equivalent to) $\perp, \mathbf{t}, \mathbf{f}$ where \perp denotes divergence, we would have a divergent computation instead of one evaluating to \mathbf{t} .

This idea was studied in the context of models of LL in [Gir99, AM99], and was given the following syntactical representation in [HvG03]. The link with Gustave function lies in what leaves are chosen by the linkings. In fact if we assign \mathbf{t} to the left premise of the $\&$, \mathbf{f} to the right one and \perp to the right one of the \oplus , then the three top linkings correspond to

$$\{ (\mathbf{t}, \mathbf{f}, \perp), (\perp, \mathbf{t}, \mathbf{f}), (\mathbf{f}, \perp, \mathbf{t}) \},$$

i.e. the trace of G (omitting the output). The other two linkings are there to satisfy the resolution condition.

The Gustave MALL net is not correct. In fact taking the correctness graph of the first three linkings gives the following cycle covering all $\&$ s, which does not even use jumps.



However γ is **Coh**-correct. Given any two experiments ϵ and f , if they are on the same simple net they are coherent because of Theorem 3.16. If on the other hand they are taken on different slices, say $\lambda, \mu \in \gamma$, then one of the three $\&s$ is binary, say $w \in \&2(\lambda, \mu)$. We then have $\epsilon(w_0) \frown f(w_0)$, which implies $\epsilon(w_0).1 \frown f(w_0).1$ on the corresponding conclusion, which in turn gives strict coherence on the whole of the sequent.

The Gustave function is rejected by Bucciarelli and Ehrhard's *strongly stable* model [BE91], from which Ehrhard developed in [Ehr95] a new model of LL, *hypercoherent spaces*, which we present up next. One turns to such a model hoping for a better account of LL additives.

3.3.3 Hypercoherent spaces

The idea at the base of hypercoherent spaces is to shift from graphs to *hypergraphs*, adapting the various operation thereafter.

The category **HCoh** of **hypercoherent spaces** has objects \mathcal{X} given by

- its web $|\mathcal{X}|$ is a set;
- a predicate $\circ_{\mathcal{X}}$ on $\mathcal{P}_{<\omega}^*(|\mathcal{X}|)$, the finite non-empty subsets of $|\mathcal{X}|$, called the **hypercoherence** of \mathcal{X} , which is *reflexive* in the sense that it contains the set of singletons $\mathcal{P}_{=1}(|\mathcal{X}|)$.

The statement $s \in \circ_{\mathcal{X}}$ is written $\circ s$ (\mathcal{X}), or simply $\circ s$ if no confusion is possible. As with ordinary coherence, apart from \circ , one defines the following relations, from which \circ can be in turn recovered

- strict hypercoherence $\frown := \circ \setminus \mathcal{P}_{=1}(|\mathcal{X}|)$;
- hyperincoherence $\asymp := \mathcal{P}_{<\omega}^*(|\mathcal{X}|) \setminus \frown$;
- strict hyperincoherence $\smile := \mathcal{P}_{<\omega}^*(|\mathcal{X}|) \setminus \circ$.

The states of \mathcal{X} are the **hypercliques** of \mathcal{X} i.e. $h : \mathcal{X}$ if and only if

$$\forall s \subseteq_{<\omega}^* h : \circ s \quad (\mathcal{X})$$

where $s \subseteq_{<\omega}^* h$ indicates that s is a finite non empty subset of h .

The following are the operations on hypercoherent spaces corresponding to LL connectives, giving **HCoh** the structure of a *-autonomous linear category. For reference we include the definition of the exponential modalities, though it will be not used in the present work.

Dual: $|\mathcal{X}^\perp| := |\mathcal{X}|$, and $\circ_{\mathcal{X}^\perp} := \asymp_{\mathcal{X}}$.

Multiplicatives: $|\mathcal{X} \otimes \mathcal{Y}| = |\mathcal{X} \wp \mathcal{Y}| := |\mathcal{X}| \times |\mathcal{Y}|$, and given $s \subseteq_{<\omega}^* |\mathcal{X}| \times |\mathcal{Y}|$ we set

$$\begin{aligned} \circ s \quad (\mathcal{X} \otimes \mathcal{Y}) &\iff \circ \pi_0(s) \quad (\mathcal{X}) \text{ and } \circ \pi_1(s) \quad (\mathcal{Y}), \\ \frown s \quad (\mathcal{X} \wp \mathcal{Y}) &\iff \frown \pi_0(s) \quad (\mathcal{X}) \text{ or } \frown \pi_1(s) \quad (\mathcal{Y}), \end{aligned}$$

with π_0 and π_1 the usual left and right projections. Units have again $1 = \perp$, $|1| = |\perp| = \{*\}$ and clearly $\circ_1 = \{\{*\}\}$.

Additives: $|\mathcal{X}_0 \oplus \mathcal{X}_1| = |\mathcal{X}_0 \& \mathcal{X}_1| := |\mathcal{X}_0| + |\mathcal{X}_1|$. Given $s \subseteq_{<\omega}^* |\mathcal{X}_0| + |\mathcal{X}_1|$, let $s_i := \{x \in |\mathcal{X}_i| \mid x.i \in s\}$. Then we set

$$\begin{aligned} \circ s \ (\mathcal{X}_0 \oplus \mathcal{X}_1) &\iff s_{1-i} = \emptyset \text{ and } \circ s_i \ (\mathcal{X}_i) \text{ for } i = 0 \text{ or } 1, \\ \circ s \ (\mathcal{X}_0 \& \mathcal{X}_1) &\iff s_{1-1} = \emptyset \text{ implies } \circ s_i \ (\mathcal{X}_i), \text{ for both } i = 0 \text{ and } 1. \end{aligned}$$

Note therefore that if s_0 and s_1 are both non-empty, one has $\wedge s \ (\mathcal{X}_0 \& \mathcal{X}_1)$ and $\vee s \ (\mathcal{X}_0 \oplus \mathcal{X}_1)$ regardless of the elements of s . Units are again $|\top| = |\mathbf{0}| = \emptyset$.

Exponentials: $!|\mathcal{X}| := \{u \in \mathcal{M}_{\text{fin}}(|\mathcal{X}|) \mid |u| : \mathcal{X}\}$. Given $s \subseteq_{<\omega}^* !|\mathcal{X}|$ we set

$$\circ s \ (!\mathcal{X}) \iff \forall u \triangleleft s : \circ u \ (\mathcal{X}),$$

where $u \triangleleft s$ (u is a multisection of s), means that $\forall b \in s$ we have $u \cap |b| \neq \emptyset$, i.e. u contains at least an element from every multiset in s . The why not modality is defined by $? \mathcal{X} := (!\mathcal{X}^\perp)^\perp$.

The operations defined above respect De Morgan's duality. Again, morphisms are the states of $\mathcal{X} \multimap \mathcal{Y} = \mathcal{X}^\perp \wp \mathcal{Y}$, thus **HCoh** is webbed.

From [Ehr95] we have an injection functor I mapping each coherent space \mathcal{X} to the hypercoherent space having

$$|I\mathcal{X}| := |\mathcal{X}|, \quad \circ s \ (I\mathcal{X}) \iff s : \mathcal{X}$$

which is left adjoint to functor PN mapping a hypercoherent space \mathcal{Y} to

$$|PN\mathcal{Y}| := |\mathcal{Y}|, \quad x \circ y \ (PN\mathcal{Y}) \iff \circ \{x, y\} \ (\mathcal{Y}).$$

The name PN is due to the fact that this operation decomposes in two polarity changing functors. PN preserves all the operations defined above but the exponential ones, which suffices us [Ehr95, Proposition 30].

Interpretation and Semantic correctness

The interpretation is again defined by interpreting type variables and applying the various linear constructs. Here we restrict ourselves to **MALL**, so there are no uniformity issues.

Theorem 3.19 (Soundness). *Every **MALL** proof net is **HCoh**-correct.*

Proof. We can through the Sequentialization theorem, as **HCoh** is a model of **LL**. We will however have this as a corollary of Theorem 5.3, which has a direct proof much in the style of that of Theorem 3.16. \square

Also we have the following relation with **Coh**-correctness.

Proposition 3.20. ***HCoh**-correctness is strictly stronger than **Coh**-correctness.*

Proof. Take a **HCoh**-correct **MALL** net θ , and any **Coh**-interpretation $\llbracket \cdot \rrbracket$. Then take the **HCoh**-interpretation $\llbracket \cdot \rrbracket'$ generated by $I(\llbracket \cdot \rrbracket)$. Now

$$\llbracket \theta \rrbracket = \llbracket \theta \rrbracket' : \llbracket I \rrbracket' \implies \llbracket \theta \rrbracket : PN(\llbracket I \rrbracket') = \llbracket I \rrbracket.$$

The implication comes directly from the definition of PN , while the last equality comes from the fact that PN preserves linear connectives and $PN \circ I$ is the identity.

Strictness comes from the fact that the Gustave MALL net of Figure 3.9 is not **HCoh**-correct. It suffices to take a set made by points from the top three linkings and the hypercoherence law on \oplus gives strict hyperincoherence on all three conclusions. \square

Proposition 3.21. *A MALL simple net λ is **HCoh**-correct if and only if it is **Coh**-correct. In particular on MLL nets the two notions coincide.*

Proof. One direction comes from the above result. The inverse one is obtained by canonically turning λ into an untyped MLL net λ' , by contracting all unary connectives in a single wire. If we assign to the axioms of λ' the variable type they had in λ (we recall we are speaking about η -expanded nets), it is possible that there is a mismatch on a cut. In such a case however the interpretation of λ is empty (as we get the same mismatch in experiments) and the result is trivial. Suppose therefore that λ' is typed. There is a bijection ϕ between experiments on λ and λ' by just assigning the same values to axioms. As unary additives preserve both hypercoherence and coherence, $\circ\phi(\{\mathbf{e}_1(\lambda), \dots, \mathbf{e}_k(\lambda)\})$ if and only if $\circ\{\mathbf{e}_1(\lambda), \dots, \mathbf{e}_k(\lambda)\}$, and the same for plain coherence. Therefore λ' is **Coh**-correct, and by Theorem 3.18 it is switching acyclic, which in turn by the soundness theorem implies it is **HCoh**-correct, which finally implies that also λ is such. \square

Once again, however, the answer to the first question raised in Section 2.3.2 is negative.

A first tedious obstacle is due to the set nature of MALL nets. If in fact we take any MALL proof net θ , then any $\Gamma \subsetneq \theta$ is necessarily a MALL net that does not satisfy the resolution condition. However as $\llbracket \Gamma \rrbracket \subseteq \llbracket \theta \rrbracket$, we have that Γ is **HCoh**-correct.

One way out of it is to just take nets satisfying the resolution condition to be the basic structures, as considered in [HvG03]. From the point of view of reduction this is quite unsatisfactory, as the resolution condition in itself is not stable under reduction. However we will for now leave it be, and mainly restrict ourselves to MALL nets satisfying such condition. We will anyway precisely state when we use such condition.

However it is not the resolution condition alone that gives problems. The counterexample, described in [Pag06b], is shown in Figure 3.10: it is a MALL net satisfying the resolution condition, unsequentializable as the final rule must be \otimes , while it cannot split the $\epsilon \oplus \epsilon, \epsilon^\perp$ part of the context as it depends on both $\&$ s. In fact an illegal cycle in such a structure is also shown. Notice that the cycle traverses the $\&$ s in opposite directions.

Proposition 3.22. *The MALL net δ shown in Figure 3.10 is **HCoh**-correct.*

Proof. Take any interpretation, and any $s \subseteq_{<\omega}^* \llbracket \delta \rrbracket$, given by experiments $\{\mathbf{e}_i\}$. If all experiments in s are taken on a single linking, then $\circ s$ by Proposition 3.21. If the linkings involved are three or four, then necessarily both $\&$ s above the tensor are binary, which gives strict coherence under the tensor. Suppose therefore that exactly two linkings are involved, and by symmetry we can suppose it is the top two. If a is the α^\perp typed free port, then if $\asymp\{\mathbf{e}_i(a)\}$ we get hypercoherence

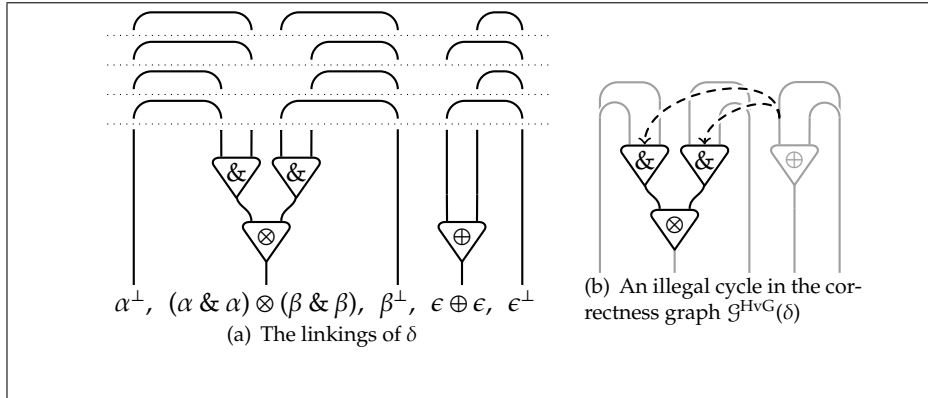


Figure 3.10: The proof structure δ : an unsequentializable structure such that $\llbracket \delta \rrbracket$ is a hyperclique.

on the left premise of the tensor, which together with strict hyperincoherence under the binary $\&$ gives strict hyperincoherence of all. \square

Notice that the right \oplus is not considered at all by the above proof. One thing that jumps to the eye is that the net is disconnected. The only way to build the cycle is then to go back and forth on the edges added by the correctness graph. It has therefore been conjectured [Pag06b, Conjecture 70] that such fracture between sequential correctness and hypercoherent semantics is due to the intrinsic unconnectedness of the counterexample.

Conjecture 3.23 (Pagani). *If θ is a cut free MALL net satisfying the resolution condition and the strong MLL one, then if θ is **HCoh**-correct it is a proof net.*

We decided to “factorize” the conjecture by first finding the criterion for semantic correctness. Such new criterion, which we call *hypercorrectness* (Definition 5.1) is for **HCoh** in MALL what visible acyclicity is for **nuCoh** in MELL and for **Fin** in DiLL (see the bottom paragraph).

More from a distance, a similarity can be established with what happened in the study of models of PCF: once it was clear that Scott-continuous functions, or even stable ones, were not fully abstract for PCF, two directions were taken. One was to refine the models (from continuity to stability and from stability to strong stability), while the other, similar to what we do here, was to find which languages were fully abstract for these same models (parallel PCF for the continuous one [Plo77] and stable PCF for the stable one [Pao06]). One difference is that in our work and that of [Pag06a, Pag08] one really finds a discerning *geometrical* criterion (something that has sense because of the presence of generally “incorrect” syntactic objects) corresponding to an *algebraic* one, apparently distant (hypercliques here, finitary relations in [Pag08]).

A Short Detour: MELL, DiLL and Visible Acyclicity

Also in MELL the inverse result of 3.16 fails. We sketch here the discussion which was developed in [Pag06a]. The idea is that while box borders are blindly seen as a sort of connected wall, coherent spaces may in some cases (but not all) consider two auxiliary ports as not connected.

The idea now is, rather than refining the semantics to fit the correctness criterion, to change the correctness criterion to fit the semantics, trying to answer the second question at the end of Section 2.3.2. In this case *visible acyclicity*, described in [Pag06a, Pag06b], is what is needed. As switching acyclicity characterizes in fact semantic correctness for **nuCoh**, the non-uniform variant (i.e. with $|\mathcal{X}| = \mathcal{M}_{\text{fin}}(|\mathcal{X}|)$) of coherent spaces introduced by Bucciarelli and Ehrhard in [BE01]³. So we have that a visible acyclic MELL net is **nuCoh**-correct and, vice versa, a cut-free **nuCoh**-correct MELL net is visible acyclic, and such criterion is stable under reduction.

Surprisingly, the result extends also to finiteness spaces **Fin**, the semantics of choice for differential linear logic. **Fin** correctness then becomes equivalent to visible acyclicity.

We will look for these kind of results in MALL and hypercoherent spaces, as explained in following sections.

³There is a technical difficulty in extending the result to **Coh**, due to the uniformity requirement.

Chapter 4

Exponential Proof Nets and Lambda Calculus

We will see in this chapter how the reintroduction of structural rules on the controlled exponential modalities brings back the expressive power of the system. This chapter is devoted to presenting exponential proof nets, both the traditional MELL (multiplicative exponential LL) ones and their extension to DiLL, differential linear logic. Particular attention is given to the results of MELL which are expected for DiLL and which will be proved in Part III.

Among these, we will present how λ -calculus translates in MELL, and how resource calculus does in differential interaction nets, which we denote by DiLL_0 .

Together with the expressive power come also non trivial reduction behaviours. Most importantly, while the linear fragments we presented in the previous chapter were confluent (in a sense strongly so) and normalizing, even without asking for correctness, as soon as boxes enter the picture this is not true anymore (see Figure 4.2).

Even with correctness in hands, proving these results is at least as hard as proving them for λ -calculus. The good news is that nets give a framework which is apt to translate many aspects of computation. So such fundamental results are apt to be then transferred to other systems. As an example, polarized proof nets by Laurent [Lau99] profit from such approach, as can, going out of LL, calculi with explicit substitutions, as shown in [CKP03].

Before going on to the main topic of the chapter, we will take the opportunity to present some advanced notion of rewriting theory that are needed to reason about reduction modulo equivalence. The proof nets we use will necessarily have such equivalence (see Figure 6.3 for a critical peak in DiLL reduction that is not joinable without associativity equivalence). Unsurprisingly, this approach works with the equivalences shown in [CKP03]. Surprisingly however we will be able to also account for what we call *bang sum* equivalence: an equivalence peculiar to DiLL derived from LL exponential isomorphism (see Figure 4.10), which splits a box containing a sum.

4.1 Rewriting Theory Modulo Equivalence

The aim of this section is making the reader acquainted with the notion of rewriting modulo equivalence, to the extent needed for our purposes. We refer to [Ter03, Section 14.3] and [Ohl98] for more indepth details and proofs. One must thread lightly, as rewriting modulo equivalence has some differences with respect to usual rewriting, the most notable being the fact the the Church-Rosser property is not equivalent to confluence. Also all confluence result are not a trivial adaptation of those regarding regular reduction, as soon as \sim is not a structural congruence, which can be described by $\sim \rightarrow \subseteq \rightarrow \sim$.

Let (S, \rightarrow) be an abstract reduction system and let \sim be an equivalence relation on S . Take also a symmetric relation \vdash such that $\vdash^* = \sim$, possibly \sim itself. Let $s \vee t$ (t and s are **joinable modulo \sim**) if $s \rightarrow \sim \leftarrow t$. We say then that \rightarrow is

- locally confluent modulo \sim if $\leftarrow \rightarrow \subseteq \vee$;
- confluent modulo \sim if $\leftarrow^* \sim \rightarrow^* \subseteq \vee$;
- locally coherent with \vdash if $\vdash \rightarrow \subseteq \vee$;
- Church-Rosser modulo \sim (or $\text{CR}\sim$) if $\approx \subseteq \vee$, where $\approx := (\rightarrow \cup \leftarrow \cup \sim)^*$;
- strongly normalizing modulo \sim (or $\text{SN}\sim$) if \rightsquigarrow is SN, where $\rightsquigarrow := \sim \rightarrow \sim$;
- strongly Church-Rosser modulo \sim if $\overleftrightarrow{\sim}$ has the diamond property, i.e. $\sim \overleftarrow{\sim} \overleftrightarrow{\sim} \sim \subseteq \overleftrightarrow{\sim} \sim \overleftarrow{\sim}$;

The last definition is our terminology, while the rest follows [Ter03]. Notice that such definitions are stronger, and more useful, than the various forms of confluence of \rightsquigarrow . The latter require to use conversion during the reductions to join a divergent peak, whereas for the former equivalence is only at the end.

Being Church-Rosser modulo \sim is the most important property of all those concerning confluence. In particular it implies the unique normal form modulo \sim property, that is that \approx coincides with \sim on normal forms, which again implies that in order to compute the normal form one can use just regular reductions, without ever be forced to \sim -convert in order to get the result.

Contrary to what happens in regular reduction, $\text{CR}\sim$ is strictly stronger than plain confluence in general, with the following ARS being a quick example



where equivalences are denoted by arrowless edges. We have however the following condition for equivalence between the two.

Lemma 4.1 (Huet, [Hue80]). *If \rightarrow is WN, then \rightarrow is $\text{CR}\sim$ if and only if it is confluent modulo \sim .*

The following instead is an important generalization of Newman's lemma in this setting.

Lemma 4.2 (Huet, [Hue80]). *If \rightarrow is $\text{SN}\sim$, locally confluent modulo \sim and locally coherent with \vdash , then it is $\text{CR}\sim$.*

Notice the quite strong hypothesis of the reduction being strongly normalizing modulo \sim , which cannot be dropped lest one incurs in a counterexample. Huet in the same work proved another version for the same result with different hypotheses: plain SN, again local confluence modulo, but local coherence must be checked with the full relation \sim , severely complicating the reasoning by critical pairs.

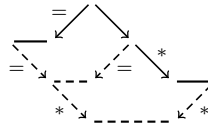
The following instead is an observation by van Oostrom.

Lemma 4.3. \rightarrow is $\text{CR}\sim$ if and only if $\overset{*}{\rightarrow}$ is strongly $\text{CR}\sim$.

The following is a result we did not find in the literature, but is useful to prove the Church-Rosser modulo property using parallel reduction.

Lemma 4.4. If \rightarrow is strongly $\text{CR}\sim$, then it is $\text{CR}\sim$.

Proof. First we prove that $\sim \overset{\equiv}{\leftarrow} \overset{*}{\rightarrow} \sim$ is joinable, by induction on the length of the reduction on the right. In case it is zero it is a direct application of strong $\text{CR}\sim$ (hence we use a reflexive closure). So suppose the length is greater than zero. In the following we draw equivalence by arrowless edges. We have the following confluence diagram.



The top hexagon is from strong $\text{CR}\sim$, while the bottom heptagon is by induction hypothesis.

Now we move on to prove that $\overset{*}{\rightarrow}$ is strongly $\text{CR}\sim$, concluding by the previous lemma. Again, we reason by induction on the length of one of the two branches. If it is zero, joinability can be recovered from the previous step. Otherwise we have a confluence diagram that is identical to the previous one, save for substituting $\overset{*}{\rightarrow}$ for the top $\overset{\equiv}{\leftarrow}$. \square

4.2 The Exponentials

In this section we will redirect our attention to another fragment of LL. While leaving aside the additives, we move to MELL, by reintroducing in MLL the structural rules. These rules correspond from the computational point of view to the duplication and erasure of resources. In the first subsection we present the system, while in the second we show the strong link with λ -calculus which is the starting point of some of the results in Part II.

Clearly the first potentialities of this fragment were studied in [Gir87]. Later developed by Danos and Regnier in their theses [Dan90, Reg92], giving a particular emphasis to their relation with λ -calculus (which is the basis for the second subsection). In particular Danos developed an alternative syntax which abstracted away trees of contractions and weakening and their interleaving with box borders. From a categorical point of view, this syntax quotients associativity of contraction and neutrality of weakening with respect to contraction, together with equations concerning the relations between the two and the exponential comonad !.

From the point of view of the calculus, weakenings correspond to the introduction of dummy variables, and contractions to the identification of two occurrences of the same variable. These usually do not correspond to actual constructs of the syntax. The equations hinted above can therefore be seen as corresponding to the fact that there is no precise “place” in the programs where variables are explicitly introduced as dummy, or identified. Seen dually, there is not, often, explicit constructs to discard or duplicate inputs.

Later Di Cosmo and Kesner introduced a variant of MELL in [DCK97] that was aimed at extending the pairing between proof nets and λ -calculus to λ -calculi with explicit substitutions. In this variant of proofnets such equations are dealt with using actual reductions.

In other works by Di Cosmo and Guerrini [DCG99], and again the first author with Kesner and Polonovski [CKP03] introduce yet another approach, using real equivalences. A little nuisance is that the equivalences concerning weakenings cannot be implemented, lest one get infinite reductions (see Remark 4.9). So it is turned into a reduction. However the real problem is that proving normalization modulo such equivalences is quite delicate.

We will want to deal with differential proof nets, and we will show during the proof of 6.13 that at least associativity cannot be ignored, lest we loose confluence (Figure 6.3). Thus we would want to apply one of the above approaches.

The first approach seems (for now) impracticable. We will get back to this when we will have introduced the rules of differential MELL in Section 4.3. Among the other two, we initially used the second in [Tra08b], but we will here move on to the more elegant third one, introducing. In order to do so we need however to introduce the basic notions of rewriting modulo equivalence. We will also take the opportunity to state the lemmas we need to prove the results in Chapter 6.

MELL types are defined by the grammar

$$\mathcal{T}_{\text{MELL}} ::= \mathcal{V} \mid \mathcal{V}^\perp \mid 1 \mid \perp \mid \mathcal{T}_{\text{MELL}} \otimes \mathcal{T}_{\text{MELL}} \mid \mathcal{T}_{\text{MELL}} \wp \mathcal{T}_{\text{MELL}} \mid !\mathcal{T}_{\text{MELL}} \mid ?\mathcal{T}_{\text{MELL}},$$

with $(!A)^\perp = ?A^\perp$. The two new modalities are indeed called **exponential**. MELL (multiplicative exponential linear logic) is the single-threaded box signature which adds to MLL the cells and the axiomatic typing rules shown in Figure 4.1. Below them the **exponential** reduction rules are shown.

An important point is that, contrary to (even incorrect) MLL or MALL nets, MELL nets are neither confluent nor normalizing, as shown by the counterexamples in Figure 4.2. We have to state the correctness criterion to get such results.

4.2.1 Correctness and Properties of Reduction

Though also for MELL there is a sequent calculus and a notion of sequentialization, it is beyond the scope of the present work. However, as already seen in the previous section, the correctness criterion is paramount for having good properties of the rewriting system. For this we can completely disregard the connectedness part of the criterion, and concentrate on switching acyclicity.

We thus have to extend the criterion stated for multiplicatives in Section 3.1. This is done with an inductive definition. Intuitively, after correctness is checked inside boxes, their contents are completely disregarded, and correctness is checked at depth zero considering the boxes as simple nodes.

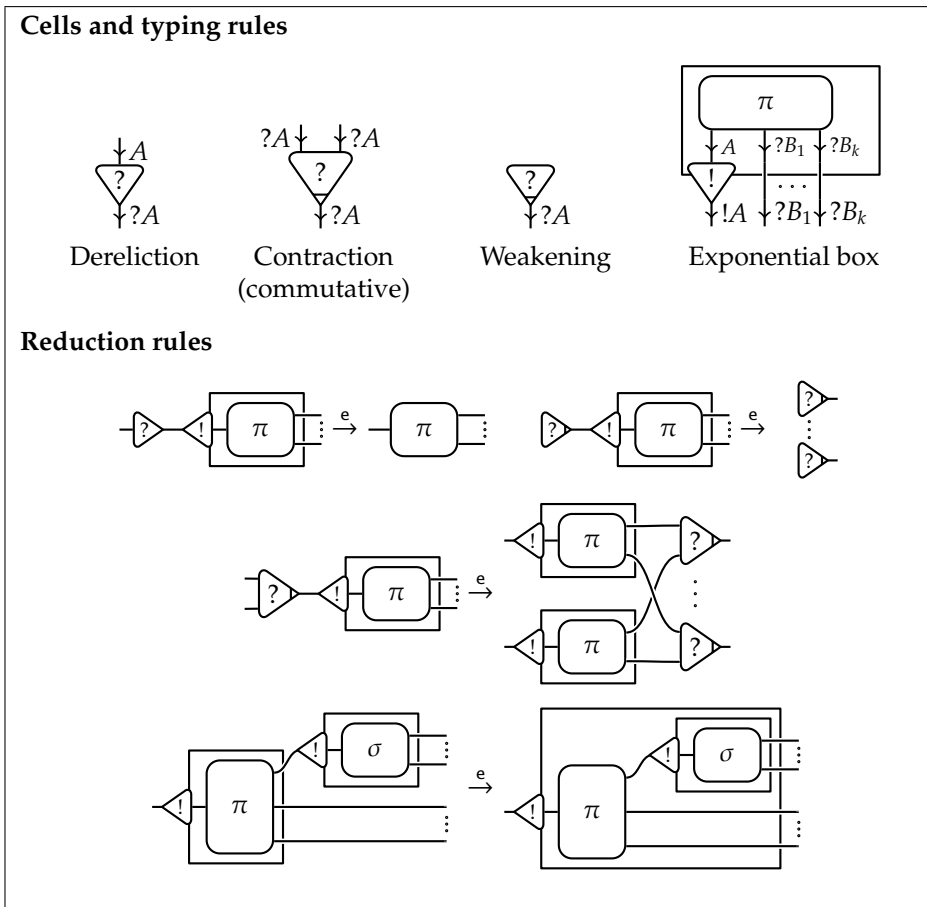


Figure 4.1: Cells of MELL, together with typing rules for cells and for the box and the reduction rules. Recall that in the box on box reduction the two boxes must be different (a condition that is not trivial in incorrect MELL nets).

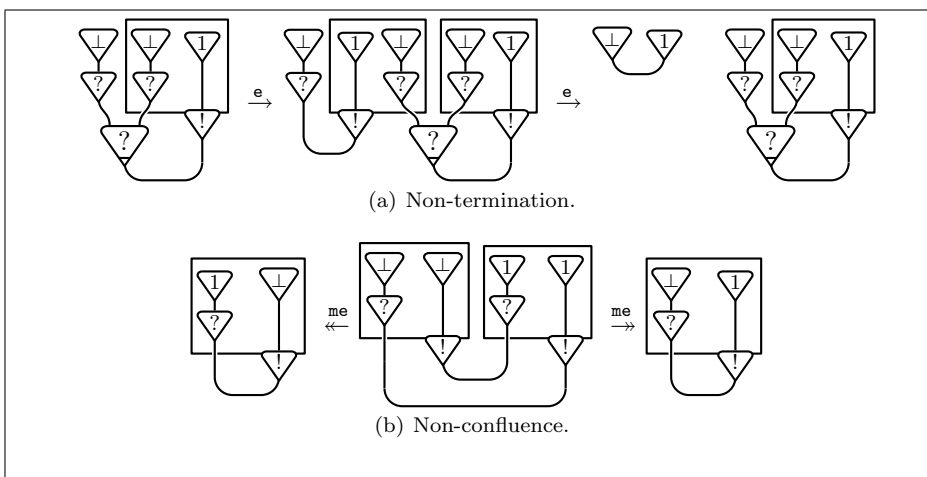


Figure 4.2: Examples of non-termination (a), and non-confluence (b). Multiplicative units are not really necessary, but simplify the examples.

We have the following formal definition, after we set as switching symbols both pars and contractions.

Definition 4.5 (Danos-Regnier correctness for MELL). A MELL net λ is said to be correct if

- for each $b \in \mathbf{b}_0(\lambda)$ box, its contents $\sigma(\lambda)$ is correct, and
- for every switching S on λ , the graph $\mathcal{G}_S(\lambda)$ is acyclic, or equivalently there are no switching cycles in $\mathcal{G}(\lambda)$.

Such definition is carried out by induction on the depth of λ .

As usual “correct net” and “proof net” will mean the same thing. Clearly an MLL+mix proof net is also a MELL one.

Theorem 4.6 (Confluence and normalization). *The reduction $\xrightarrow{\text{me}}$ on MELL proof nets is confluent and strongly normalizing.*

The above result is practically contained in Danos’ thesis [Dan90]. For another published proof, the reader is referred to [PTdF08], where also additives are accounted for.

In reality one can generalize the above results in two direction. An untyped MELL net which satisfies the correctness criterion is called a **MELLpure** proof net. We cannot hope for termination as we will see that a fragment of pure proofnets encodes pure λ -calculus. We have the following results.

Theorem 4.7 (Confluence of pure proof nets). *MELL pure proof nets are confluent under the reduction \mathbf{r} .*

Let $\xrightarrow{\text{er}}$ be the subset of reductions without the case weakening against box. This reduction is called **non erasing**: it is the only reduction that can destroy other redexes, the ones inside the box.

Theorem 4.8 (Conservation). *For λ a MELL pure proof net, $\lambda \in \text{WN}_{\text{er}} \iff \lambda \in \text{SN}_{\text{me}}$.*

Said in other words, if $\lambda \xrightarrow{\text{er}} \mu$ and $\mu \in \text{SN}_{\text{me}}$, then $\lambda \in \text{SN}_{\text{me}}$. If we find a strategy reducing a net without ever erasing, until only erasing steps are available, then the net is strongly terminating.

The nomenclature refers to the analogous theorem of λ -calculus proved by Church and later by Barendregt, Bergstra, Klop and Volken[Bar84]. For MELL it has been called *striction lemma* in [Dan90]. This result is the ideal launching pad for results of normalization. For example [PTdF08], which is the first complete proof of strong normalization for full second order linear logic proves and uses such result, completing it with a proof of non erasing weak normalization by candidates of reducibility.

4.2.2 Accommodating the Contractions: Associativity, Neutrality, Push and Pull

We now introduce the equivalences, dealing with associativity and commutation of contractions with respect to box borders. For each of these two equivalences

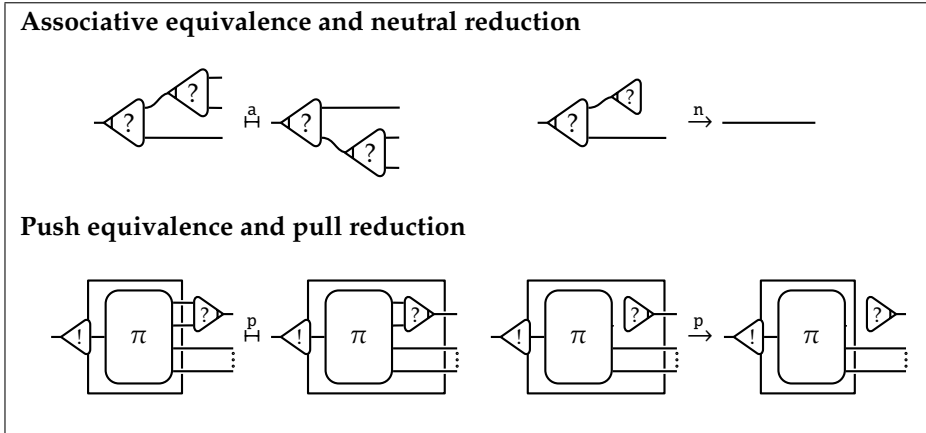


Figure 4.3: The generating pairs for associative and push equivalences, and the neutral and pull reductions.

we have corresponding *generalized reduction rules* for weakenings. The **associative** and **push** equivalences $\overset{\sim}{\sim}$ and $\overset{\sim}{\sim}$, together with the **neutral** and **pull** reductions \mathbf{n} and \mathbf{p} are shown in Figure 4.3. The full relations $\overset{\sim}{\sim}$ and $\overset{\sim}{\sim}$ are defined by context and symmetric closure of the ones shown, while the full equivalences $\overset{\sim}{\sim}$ and $\overset{\sim}{\sim}$ are generated by transitive reflexive closure of their \vdash versions. We will call \mathbf{c} (**canonical** reduction) either the \mathbf{p} one alone, or the \mathbf{np} one, union of \mathbf{n} and \mathbf{p} , the one, denoted by \mathbf{c} . Similarly, we will simply denote by \sim either $\overset{\sim}{\sim}$ or $\overset{\sim}{\sim}$.

Remark 4.9. Neither \mathbf{n} nor \mathbf{p} can be turned into equivalences. Figure 4.4 shows examples of infinite \mathbf{e} reductions once we are allowed to reverse one of the two. The inherent reasons for this can be explained. The reversal of \mathbf{n} can clearly turn any normal form (at least one having an exponentially typed wire) in a redex. For \mathbf{p} the reason is more subtle: such rule would connect parts otherwise unconnected, introducing more possible reduction. The example of Figure 4.4 show that even if we restrict the reduction only when it does not introduce switching cycles, still this new connectedness can be dangerous on the long run.

Theorem 4.10 (normalization of \mathbf{me} modulo $\overset{\sim}{\sim}$, [CKP03]). *The reduction $\overset{\sim}{\sim}$ on MELL proof nets is strongly normalizing modulo $\overset{\sim}{\sim}$.*

It must be noted that in MELL the main interest in considering the \mathbf{ap} -equivalence and the \mathbf{c} -reduction is the strong relation with λ -calculi with explicit substitutions. This is a hint for a possible future exploitation of the results presented in Part III. In any case, \mathbf{a} -equivalence and \mathbf{n} -reduction are definitely needed for confluence in DiLL.

4.2.3 λ -nets and λ -calculus

In this section we explore the link between proof nets and λ -calculus. We give two versions of the system which we call λ MELL, a typed and untyped one. The first, typed λ MELL, is none other than the image of the \rightarrow fragment of

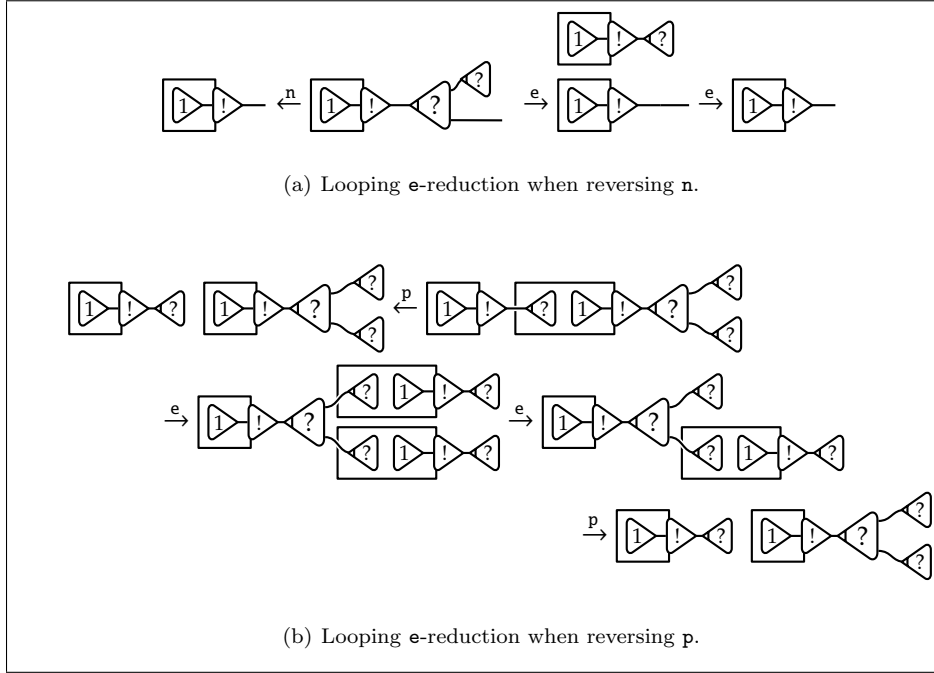


Figure 4.4: Examples of infinite reductions modulo \simeq and \simeq^{\sim} . The use of the multiplicative unit 1 simplifies the counterexamples but is not required.

intuitionistic logic into intuitionistic linear logic via Girard's translation:

$$X^{\circ} := X, \quad (A \rightarrow B)^{\circ} := !A^{\circ} \multimap B^{\circ}, \quad (\Gamma \vdash A)^{\circ} := !\Gamma^{\circ} \vdash A^{\circ}.$$

The second, to which we will usually refer simply as λMELL , is not untyped in the sense of proof nets, but rather as providing the framework for untyped λ -calculus. We still need some notion of type in order to avoid clashes (which in λ -calculus cannot occur).

The Nets

Let us give the following two mutually recursive grammars of **anti-output** types \mathcal{J} and **output** types \mathcal{O} :

$$\mathcal{J} ::= \mathcal{V}^{\perp} \mid !\mathcal{O} \otimes \mathcal{J}, \quad \mathcal{O} ::= \mathcal{V} \mid ?\mathcal{J} \wp \mathcal{O}.$$

Let us moreover call types of the form $?\mathcal{J}$ **input** types, and those of the form $!\mathcal{O}$ **anti-input** types. The types for λMELL are the subset of MELL types obtained by restricting types to input, output, anti-input and anti-output ones.

In light of Girard's translation of the intuitionistic arrow in linear logic, the grammar for output types is seen as being

$$?\mathcal{J} \wp \mathcal{O} = !\mathcal{O} \multimap \mathcal{O} = \mathcal{O} \rightarrow \mathcal{O},$$

that is the grammar for the types of (first order) λ -calculus.

Types for pure λMELL are the four types ι (anti-output), $?\iota$ (input), o (output) and $!o$ (anti-input), with duality set to $\iota = o^{\perp}$ and clearly $(\perp!o) = ?\iota$.

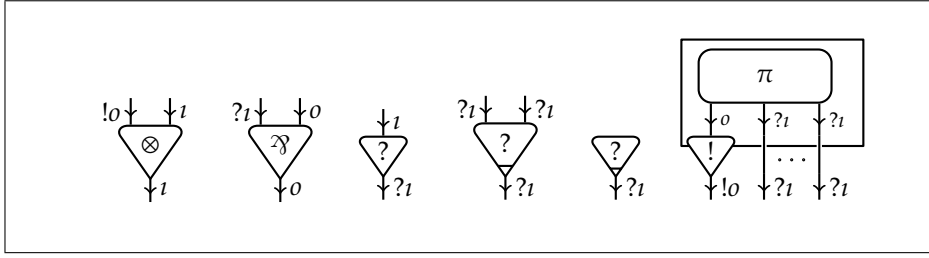


Figure 4.5: Typing rules for pure λ MELL.

These types can be seen as the quotient of the above ones with respect to identifying all variables and moreover

$$o \rightarrow o \equiv o, \quad \text{i.e.} \quad ?l \otimes o \equiv o,$$

extended by compatibility with duality. This is exactly what is needed to “type” pure λ -calculus.

A **typed λ -net** μ is a MELL proof net where all types are in λ MELL¹, that moreover satisfies the following constraints:

$\lambda 1$) for every switching S the graph $\mathcal{G}_S(\mu)$ has a number of connected components equal to the number of weakenings in $c_0(\mu)$ plus one; every box contents has inductively the same property;

$\lambda 2$) no input formula appears in $\tau_\mu(\mu)$.

Pure λ -nets (or simply λ -nets) are obtained from the same cells of typed λ MELL, with the type system of pure λ MELL and the typing rules depicted in Figure 4.5. These typing rules are the usual ones quotiented by the equivalence classes of pure λ MELL. Again one requires correctness (i.e. switching acyclicity) and the two points above.

Every typed λ -net can be easily mapped to a pure λ -net, by canonically mapping each type to its equivalence class. So in fact all results proved for pure λ -nets are valid also for the typed version.

It is known in graph theory that given a graph $G = (V, E)$ with nodes V and edges E , if we define by $\#G$ the number of connected components of G , then

- if G is acyclic, then $\#V = \#E + \#G$;
- viceversa if $\#V = \#E + k$, then G is acyclic iff $\#G = k$.

As for all switchings the number of nodes and the number of edges of $\mathcal{G}_S(\mu)$ is the same, it follows that we can equivalently substitute point $\lambda 2$ above with a version that checks only a single arbitrarily chosen switching. Moreover it turns out that the conditions stated above implicitly shape the sequent in an intuitionistic form, as explained by the following result.

Proposition 4.11 ([Dan90]). *A λ -net has exactly one conclusion typed either with o or with $!o$.*

In fact more in general, we have the following property.

¹We therefore leave out the constants 1 and \perp .

Proposition 4.12. *A switching acyclic net μ with types in λMELL (either typed or pure) enjoys $\lambda 1$ and $\lambda 2$ if and only if $\tau(\mu) = ?\iota, \dots, ?\iota, o$ or $\tau(\mu) = ?\iota, \dots, ?\iota, !o$.*

Proof. Let us take a *principal* switching S , that is a switching that for every \mathfrak{A} chooses the o typed port. Because of switching acyclicity the graph $\mathcal{G}_S(\mu)$ is a forest. Let w be the number of weakenings of μ , and k the number of $o/!o$ conclusions. Let us show that $\#\mathcal{G}_S(\mu) = w + k$, which concludes the proof. More precisely we show that every connected component contains exactly one weakening or $o/!o$ conclusion.

For uniqueness, let us consider an elementary directed path ϕ in $G_S(\mu)$ starting with a directed edge d with $\tau(d) = \iota$ or $?\iota$. By definition of principal switching one sees that all wires in ϕ are also typed ι or $?\iota$. It follows that a connected component of $\mathcal{G}_S(\mu)$ contains at most one node which is a weakening or a $o/!o$ conclusion, as there cannot be a path between two such nodes. For existence, in every connected component let us take any maximal straight path which traverses tensors through the linear ports. All directed wires in it are typed the same, so one extremity must be typed as o or $!o$, and the only possibility is that such extremity is a conclusion or a weakening. \square

The Calculus and the Translation

Alonzo Church's λ -calculus is sufficiently standard to skip over most of the details. The main references are [Bar84] and [Lam92]. Terms are defined by the grammar

$$A ::= \mathcal{P} \mid \lambda \mathcal{P}.A \mid (A) A,$$

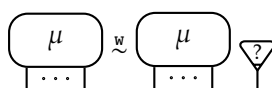
where the three constructs are **variables**, **abstractions** and **applications**. As can be seen we are using Krivine's notation, and reusing the set of ports of nets as the set of variables, as it will come in handy when we will do the translation up next. The α -equivalence and the substitutions, denoted by $u[x := v]$ for substituting v for x in u , are defined as usual. We use the notation $x \in u$ to denote that x is a free variable in u . β -reduction, denoted by $\xrightarrow{\beta}$, or simply \rightarrow , is defined by the usual context closure of

$$(\lambda x.r) s \xrightarrow{\beta} r[x := s].$$

We present here the mapping from pure λ -terms to λ -nets, as thoroughly studied by Danos and Regnier in [Dan90, Reg92].

They use a syntax for nets quotienting associativity of contraction, neutrality of weakening and commutation of contractions and weakenings with respect to box border. This quite exactly corresponds to the behaviour of λ -calculus. As we will extend the translation to differential λ -nets and resource calculus, and we are not yet able to adopt such a syntax, we use Di Cosmo and Guerrini's approach, as already explained.

In order to proceed, we first introduce an equivalence between λ -nets equating two nets differing only by conclusions directly below weakenings. That is, we adopt the smallest equivalence relation containing



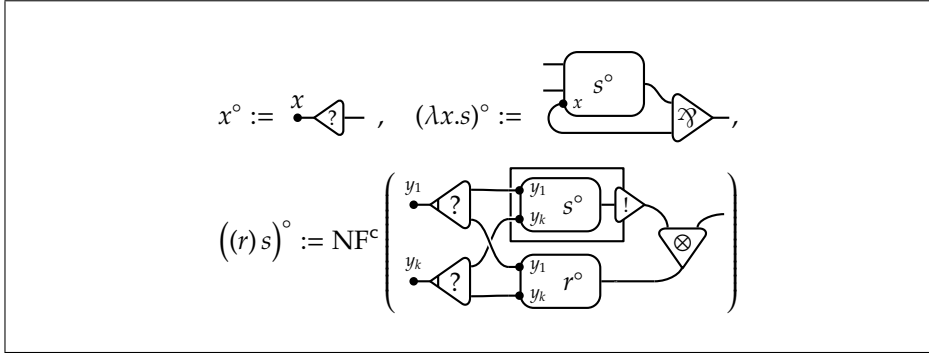


Figure 4.6: Inductive rules for the definition of t° . Labels give names to ports, which are identified with the variables of the translated term. Conversion modulo $\overset{\circ}{\sim}$ is implicitly used.

In this way, any net μ can be given any set $\text{fp}(\mu)$ of ports, provided it contains the non weakened ones. Notice that we *do not* close $\overset{\circ}{\sim}$ by context.

The translation comes in two flavours, t° and t^\bullet . The first is an exact representation of the terms, the second quotients terms with an operational equivalence. The inductive definition of the translation t° is shown in Figure 4.6. Here are some observations.

- We label the ports with their name, identified with the name of the variable to which they correspond.
- In the typed case the translation is in fact none other than the desequentialization of the translation into intuitionistic linear logic of the natural deduction proofs behind the terms.
- The equivalence $\overset{\circ}{\sim}$ is implicitly used. In abstraction we thus introduce x with a weakening if it is not among the ports of s° . In application we add enough ports to both r° and s° to unify them.
- The canonical normalization in the application steps does none other than pull the weakenings from the box containing the argument s° and turning the corresponding contractions into wires with neutrality. This ensures that the translation is well-defined with respect to $\overset{\circ}{\sim}$: any weakenings we add to s° do not change the resulting λ -net.
- The box in which one places the argument in application denotes the fact that while the function position is linear, the argument one is not. This will be seen more in detail in the resource λ -calculus.

Here are the properties of the t° translation.

Theorem 4.13 (Bijective sequentialization). *The translation t° is bijective on ec-normal λ -nets with no exponentially typed axiom and with an o conclusion, modulo $\overset{\circ}{\sim}$ and \sim .*

The condition on axioms ensures that no application to a variable is hidden by a sort of η -contracted axiom. The one on the conclusion is to avoid translations of terms inside boxes.

Theorem 4.14 (One-step bisimulation). $s \xrightarrow{\beta} t$ if and only if $s^\circ \xrightarrow{\text{m}}^{\text{ec}} \sim t^\circ$.

Theorem 4.15 (Simulation). $s \xrightarrow{\beta^+} t$ if and only if $s^\circ \xrightarrow{\text{mec}^+} \sim t^\circ$.

These theorems will be reproved in the context of differential nets and full resource calculus, in Chapter 8.

The second translation (the one truly presented in [Dan90]), is simply obtained by

$$t^\bullet := \text{NF}^{\text{m}}(t^\circ).$$

This destroys a certain degree of sequentialization between redexes. Regnier introduced in [Reg94] the equivalence relation corresponding to the identification of the terms with the same \bullet translation. However we will not yet introduce it here, leaving it for future work. The t° translation however proves to already be a potent tool.

4.3 Differential Nets, Differential λ -nets and Resource Calculus

The first steps towards a differential linear logic were made in two works by Ehrhard and Regnier.

The first, [ER03], pertained to the world of λ -calculus, presenting an extension with differential operators. This provided a tool to study ordinary λ -calculus as well. In [ER06a] and [ER08] the two authors fully introduce the notion of Taylor expansion of an ordinary λ -terms, already hinted in their previous work. One fundamental tool for the Taylor expansion is the *resource calculus*, a linear revision of Boudol's λ -calculus with resources [Bou93], which we will shortly present.

The second, [ER06b], brings forth the idea of introducing syntactic differential operators in linear logic proof nets. The objects presented there, *differential interaction nets*, are promotion-free but have formal sums of nets. We therefore move (if we restrict ourselves to natural coefficients) in polynets without boxes. One of the most important achievements in this formalism has been gained by translating into it (via other process calculi) the replication-free π -calculus in [EL07]. This has gained particular attention, as it could be the starting point for extending the Curry-Howard isomorphism to concurrent and nondeterministic computation.

In this section we first outline differential interaction nets, and then briefly present the translation into them of the resource calculus. Finally, we present the full system of differential nets, the main characters of Part III.

4.3.1 Differential Interaction Nets

The system of differential interaction nets, which we call DiLL_0 (as it is in fact DiLL restricted to 0-depth nets) is obtained by depriving MELL of the box and adding two new rules for the $!$ modality, presented in Figure 4.7, together with the needed reduction rules. A DiLL_0 net is a *polynet* built from such cells.

Proposition 2.15 applies, so the system is confluent. Correctness is established by keeping \wp and $?_k$ as switching symbols. We therefore call a DiLL_0 proof net without switching cycles. We therefore have the expected theorem.

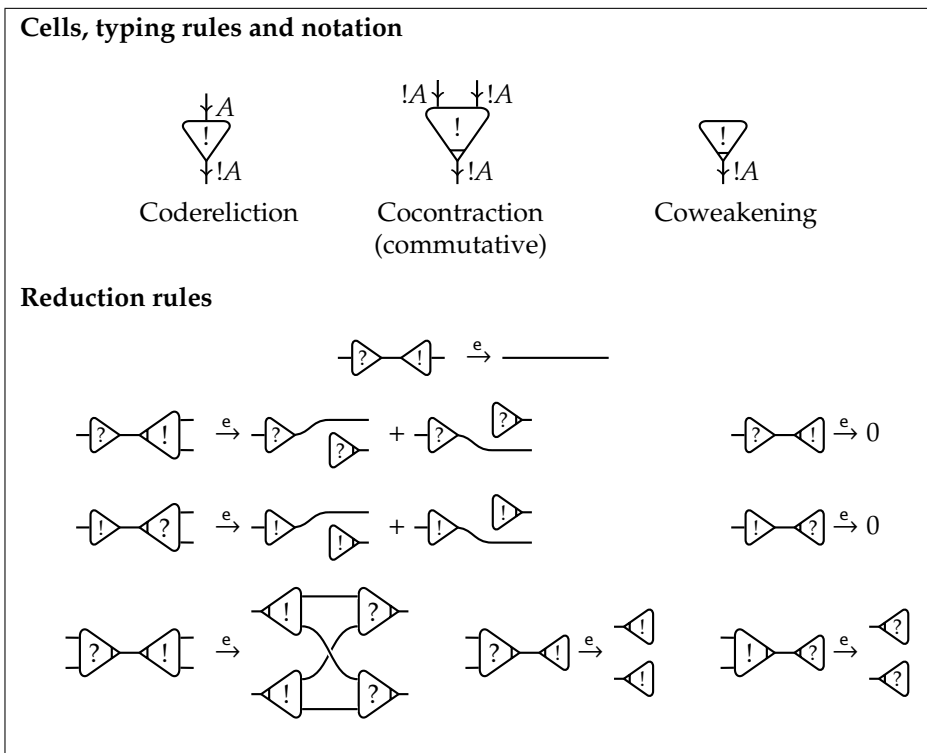


Figure 4.7: The additional cells, typing and reduction rules for differential interaction nets, DiLL_0 . Rules for ! and ? are totally symmetric.

Theorem 4.16. *DiLL₀ proof nets are confluent and strongly normalizing under the me reduction.*

In [ER06b] the \mathbf{a} -equivalence relation is proposed (though not investigated) to handle associativity and neutrality. Here we extend the \mathbf{a} -equivalence and the \mathbf{n} -reduction to cocontractions, in the trivial way. All our results for DiLL° about normalization or confluence will be done modulo at least \approx , and thus such results seamlessly apply to DiLL_0 .

4.3.2 Resource Calculus

Resource calculus is a fragment of differential λ -calculus [ER03] that is very similar to the fragment of Boudol's λ -calculus with resources [Bou93] without infinitely available resources. This calculus is seen in later works by Ehrhard and Regnier [ER08, ER06a] as the target language of the Taylor expansion of ordinary λ -calculus.

This language places itself in a small streak of calculi where the use of arguments is limited. Apart from the already cited λ -calculus with resources [Bou93], the same author in the same work also defines the fragments with multiplicities, and one can see further developments on the same calculus or a slightly different calculus in [BCL99, Kfo00].

Resource calculus has in fact the same syntax of Boudol's calculus, without u^∞ arguments. The main differences lie in the reduction. Firstly, non-determinism is accounted for by means of formal sums. Secondly, the reduction is not restricted to weak head reduction. This two differences are in fact fully inherited from differential λ -calculus.

The set Δ of simple terms of resource calculus is defined by the grammar

$$\Delta := \mathcal{P} \mid \lambda \mathcal{P} . \Delta \mid \langle \Delta \rangle \Delta^!$$

where $\Delta^! := \mathcal{M}_{\text{fin}}(\Delta)$ is the set of **bags of arguments**, or simply bags². Bags are presented in multiplicative notation. **Differential terms**, or simply terms, of resources calculus are again finite multisets of simple terms, i.e. $\mathbb{N}\langle \Delta \rangle$. Though terms and bags are in fact the same, we distinguish between them as they have a fundamentally different role. Terms are therefore presented in additive notation. Analogously, we write of **differential bags** for $\mathbb{N}\langle \Delta^! \rangle$. We write $\Delta^{(!)}$ (resp. $\mathbb{N}\langle \Delta^{(!)} \rangle$) when we do not want to specify whether we are dealing with simple terms or bags (resp. terms or differential bags).

The constructors are all extended by multilinearity to $\mathbb{N}\langle \Delta^{(!)} \rangle$. So

$$\langle u + v \rangle A = \langle u \rangle A + \langle v \rangle A, \quad (u + v)A = uA + vA, \quad \langle s \rangle (A + B) = \langle s \rangle A + \langle s \rangle B. \quad (4.1)$$

As usual by $x \in t$ for $t \in \Delta^{(!)}$ or $t \in \mathbb{N}\langle \Delta^{(!)} \rangle$, we mean that x appear free in (any simple term in) t . Of the ordinary substitution of λ -calculus that substitutes all occurrences of a variable, only the special case $u[x := 0]$ remains, which can be defined in a single step by

$$u[x := 0] := \begin{cases} u & \text{if } x \notin u, \\ 0 & \text{otherwise.} \end{cases}$$

²They are called polyterms in the works by Ehrhard and Regnier, but we have decided to follow Boudol's terminology to avoid confusion with the notion of polynet that has a different meaning.

Its role is taken by **linear substitution** $\frac{\partial u}{\partial x} \cdot v$ that substitutes v in u for a single occurrence of x . u is a simple term, v is a differential one and the result is differential. The inductive definition is as follows

$$\begin{aligned} \frac{\partial y}{\partial x} \cdot v &:= \delta_{y,x}v, & \frac{\partial \lambda y.s}{\partial x} \cdot v &= \lambda y. \left(\frac{\partial s}{\partial x} \cdot v \right) \quad \text{where } x \neq y \text{ and } y \notin v, \\ \frac{\partial \langle r \rangle A}{\partial x} \cdot v &:= \langle \frac{\partial r}{\partial x} \cdot v \rangle A + \langle r \rangle \left(\frac{\partial A}{\partial x} \cdot v \right), & \frac{\partial A}{\partial x} &:= \sum_{u \in A} \left(\frac{\partial u}{\partial x} \cdot v \right) A/u. \end{aligned}$$

The clause for abstraction can always be achieved by α -conversion. The “quotient” A/u is the multiset A without u . In fact the addends of the last sum are just obtained by substituting each of the terms in the bag with its linear substitution.

The linear substitution can be viewed, how the notation suggests, as the differential of u with respect to the variable x , linearly applied to v . This is hinted by the rule for linear application, which relates to the rule for composition of the differential. Also we have that if $x \notin u$, i.e. u is constant with respect to x , then $\frac{\partial u}{\partial x} \cdot v = 0$. This intuition is furtherly strengthened by the validity of the *Schwartz lemma*, as if $x \notin u$ and $y \notin v$ then

$$\frac{\partial}{\partial x} \left(\frac{\partial s}{\partial y} \cdot v \right) \cdot u = \frac{\partial}{\partial y} \left(\frac{\partial s}{\partial x} \cdot u \right) \cdot v.$$

In particular, for an iterated differential with respect to a variable x

$$\frac{\partial}{\partial x} \left(\dots \left(\frac{\partial s}{\partial x} \cdot u_1 \right) \dots \right) \cdot u_k$$

the order of the u_i is irrelevant if $x \notin u_i$ for all i . Then we are allowed to write

$$\frac{\partial^k s}{\partial x^k} \cdot (u_1 \dots u_k)$$

with $u_1 \dots u_k$ a bag of arguments A with $x \notin A$.

We are now ready to define the reduction, which comes in two flavours.

Baby-step reduction $\xrightarrow{\text{bs}}$ is defined by the context closure of

$$\langle \lambda x.s \rangle (uA) \xrightarrow{\text{bs}} \langle \lambda x. \frac{\partial s}{\partial x} \cdot u \rangle A, \quad \langle \lambda x.s \rangle 1 \xrightarrow{\text{bs}} s[x := 0].$$

The intuitive idea is that this reduction takes one resource from the bag at a time, performing a single linear substitution at a time, leaving the redex behind. When a bag is exhausted, we reduce to zero if the bounded variable is still present, or erase the redex if it is a K -redex, i.e. a redex with no occurrence of the bounded variable.

This atomic reduction is closer to the one in Boudol’s calculus with resources. We have however two striking differences. When the resources provided are more than the occurrences of the bounded variable we have seen that we reduce to 0. This does not happen in Boudol’s calculus, as it is an *affine* one. On the other hand when less resources than needed are provided, again we here have a reduction to 0, while in Boudol’s calculus the reduction was stopped and deemed a deadlock.

Giant-step reduction $\xrightarrow{\text{gs}}$ empties a whole bag in one step. It is thus the context closure of the following step:

$$\langle \lambda x.s \rangle A \xrightarrow{\text{gs}} \frac{\partial^{\#} A_s}{\partial x^{\#} A} \cdot A [x := 0].$$

We can informally tell what the result of the above step is. If we denote by x_1, \dots, x_h all the occurrences of x in s , then

$$\frac{\partial^k_s}{\partial x^k} \cdot (u_1 \dots u_k) [x := 0] = \begin{cases} 0 & \text{if } k \neq h, \\ \sum_{\sigma \in S_k} s[u_{\sigma(1)}/x_1, \dots, u_{\sigma(k)}/x_k], & \text{if } k = h. \end{cases}$$

with the square brackets taking the usual role of substitution.

This calculus, even in the untyped case, is strongly normalizing under both reductions, having a unique normal form for both, so that it is also confluent.

Linear Differential λ -nets and the Translation

Following the definition of λ -nets given in Section 4.2.3, we define linear differential λ -nets, based on DiLL_0 . In fact, we do not have to add anything to the definition, and that is way we denote the nets with the same name. A typed (resp. pure) linear differential λ -net is simply a DiLL_0 proof net with types in λMELL (resp. in pure λMELL) satisfying properties $\lambda 1$ – 2 . Proposition 4.12 still holds, so equivalently we can require the proof net to have conclusion $?J, \dots, ?J, \emptyset$ or $!O$ (resp. the corresponding pure types).

We therefore translate differential terms and bags of resource calculus with the inductive rules presented in Figure 4.8, again considering differential λ -nets up to \equiv_w equivalence. Again \equiv_w conversion is implicitly used, the same way as was done for ordinary λ -calculus, and \mathbf{c} -reduction eliminates useless weakenings and coweakenings (the latter disappear from a bag as soon as it is not empty).

We here have another issue: the definition corresponds to a function only if we consider nets up to \mathbf{a} -equivalence, as otherwise we get different nets depending on the different ordering of a bag.

Notice that differential and simple terms get a translation typed with o (apart from the $?i$ conclusions), while bags of arguments get an $!o$ conclusion.

The simulation result sketched in [ER06b] is a bit imprecise, as they state the simulation of the baby-step reduction. More precisely we have real simulation for the giant-step one, as any reduction step requires a multiplicative redex to be burnt, so that we cannot simulate the progressive emptying of a bag done in the \mathbf{bs} -reduction.

Theorem 4.17. *If $s \xrightarrow{\text{gs}} t$ then $s^\circ \xrightarrow{\mathbf{m}} \xrightarrow{\mathbf{e}} \sim t^\circ$.*

In Section 8.3, we will extend the results seen in Section 4.2.3 to the full resource calculus, and therefore to resource calculus also.

4.3.3 Differential Nets and Boxes

This last section is dedicated to briefly show the syntax of DiLL proof nets. Some remarks are made, but the main and original results will be shown in Part III.

DiLL is the signature with *multi-threaded* boxes obtained from the cells and typings of DiLL_0 together with the box from MELL . The missing reduction rules

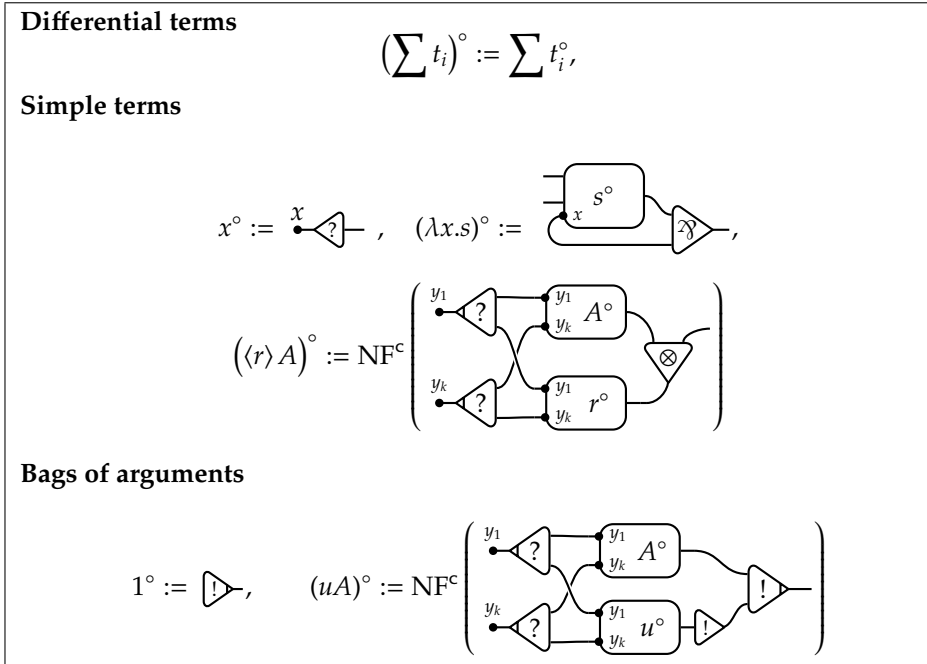


Figure 4.8: Inductive rules for the definition of t° for resource calculus.

are shown in Figure 4.9. It must be noted that due to the fact that boxes now contain sums, and the way we have defined glueing, both the dereliction against box reduction already shown in Figure 4.1 and the codereliction against box one of Figure 4.9 may in fact introduce sums, or even 0.

We report here a remark we will see in Chapter 6 when proving local confluence.

Remark 4.18. The e-reduction (and the me-reduction) is not confluent without the associative equivalence or the neutral reduction.

This is a striking difference from LL, due to the particular shape of the codereliction against box reduction, which introduces binary contractions and cocontractions. We therefore definitely have to consider the a-equivalence and the n-reduction.

The introduction of the push equivalence and the pull reduction need some particular attention in this setting where sums are found inside boxes.

Firstly, permitting p-equivalence when the box contains 0 would incur in the same problem shown in Remark 4.9 for n-reduction. A box containing 0 could spawn by equivalence an arbitrarily big contraction tree under its auxiliary ports, as 0 is equal to 0 glued to a contraction. Again there could be a net which is never normal due to a 0 box inside it. We therefore restrict the p-equivalence to boxes with non-zero contents.

Secondly, n-reducing a contraction in a single addend of a sum may break the local coherence of the reduction with respect to p-equivalence. More on this will be clear when local coherence diagrams will be checked in Section 6.3.2.

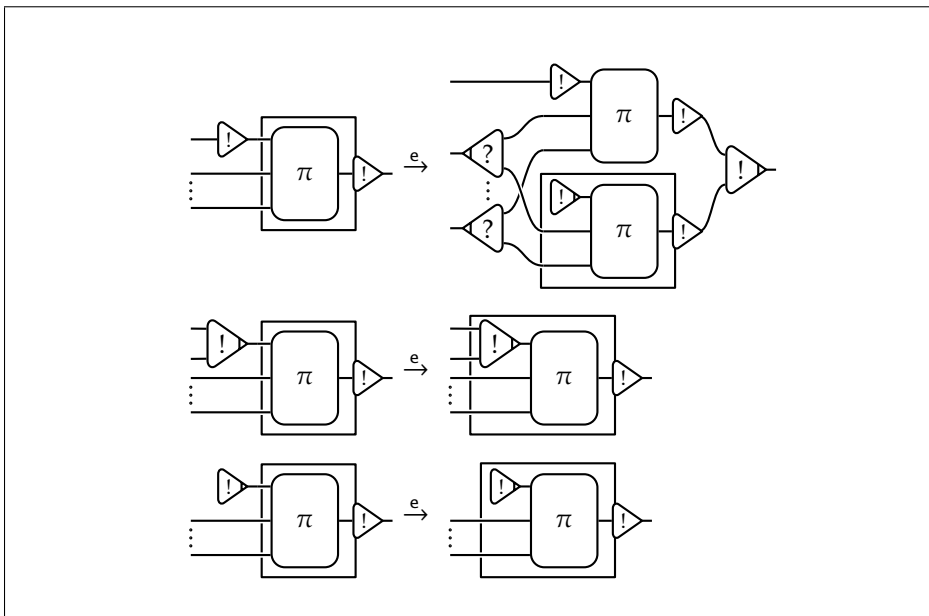
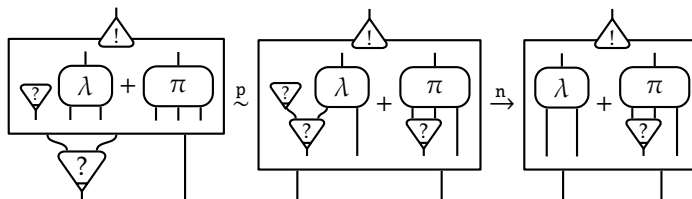


Figure 4.9: Additional exponential reduction rules for DiLL.

This can be briefly depicted by the following divergence.



In LL such local coherence is ensured by \mathbf{p} -reduction, which here is unable to join the two members on the left and right as it cannot be applied on the left. There should be a way to separate the internal sum on the left.

This introduces us to another important equivalence specific to DiLL. It equates a sum inside a box with two boxes side by side linked by a cocontraction under a principal port and contractions under the auxiliary ones. This equivalence comes in fact from the so called *exponential isomorphism* of LL $!(A \& B) \sim !A \otimes !B$, as the sum of DiLL can be modeled by a biproduct (i.e. a product and coproduct at the same time)³

The new version of the push equivalence and pull reduction are shown in Figure 4.10, together with the **bang sum equivalence** \mathbf{z} . Once more, this equivalence has a corresponding reduction dealing with a “degenerate” case, the **bang zero reduction** \mathbf{z} which turns a box containing 0 into a coweakening.

Once again, the new reduction \mathbf{z} here introduced cannot be reversed.

³This isomorphism gets in this way an even stronger relation with the equality $e^{x+y} = e^x \cdot e^y$, as the box has in fact the Taylor expansion of the exponential. Also related is the equivalence $!(P + Q) \sim !P \mid !Q$ of the π -calculus.

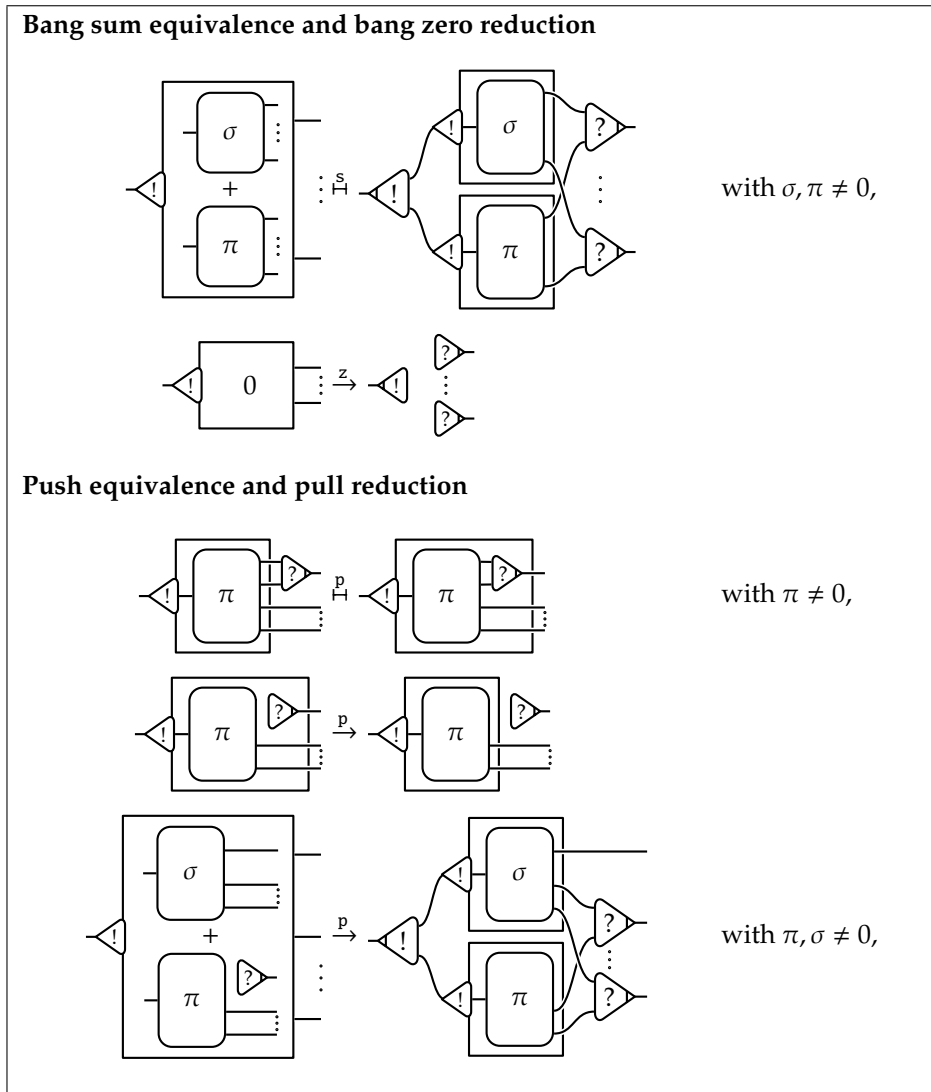


Figure 4.10: The pairs generating the push and sum equivalences, and the pull and zero reductions.

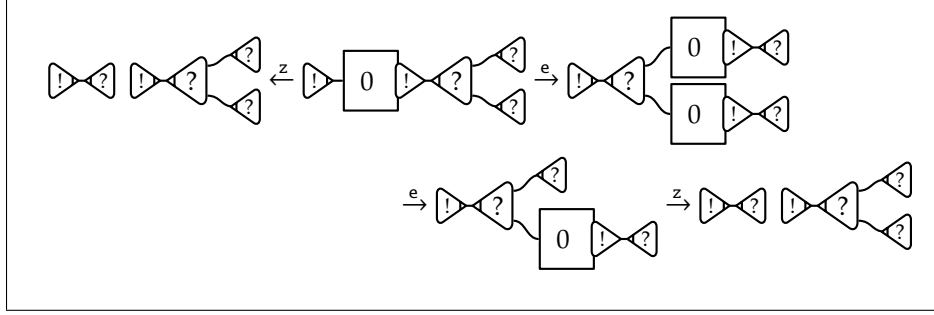


Figure 4.11: Example of looping e -reduction with the reversal of the z -reduction.

Remark 4.19. The z -reduction cannot be reversed. Once again, as for the p -reduction (see Remark 4.9), the reason lies in the creation of switching paths, and once again even if we restrict the rules only when the result is correct, we can easily get an infinite reduction. See Figure 4.11 for the example.

We would like to stress the fact that while the associative equivalence and neutral reduction are necessary for confluence, these other ones may be employed or not. The bang sum and zero ones may be used without the push and pull ones, but not viceversa.

As usual, we will denote by \sim the equivalence generated by the a , p and s ones, and by \xrightarrow{c} the canonical reduction given by n , p and z .

Remark 4.20. A good point in favour of the use of the proposed equivalences and reductions is that η -equivalence is not an observational equivalence without them, i.e. one can provide two η -equivalent forms which normalize to non- η -equivalent ones.

The following reduction shows the point.

$$\begin{array}{c}
 \text{!} \text{---} \text{!} \\
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---}
 \end{array}
 \equiv_{\eta}
 \begin{array}{c}
 \text{!} \text{---} \text{!} \\
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---}
 \end{array}
 \xrightarrow{e}
 \begin{array}{c}
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---}
 \end{array}
 +
 \begin{array}{c}
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---}
 \end{array}
 \xrightarrow{c}
 \begin{array}{c}
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---}
 \end{array}
 \quad (4.2)$$

This is interesting in itself, as it is known from finiteness spaces [Ehr05] that on the categorical level the cocontraction is $!a \circ n$ where a is the codiagonal of the biproduct $+$ and n is the exponential isomorphism. Equation (4.2) gives a syntactical evidence of this by the analog of the $\beta\eta$ -equivalence, as the categorical interpretation of the third member turns out to be just $!a \circ n$. See [Bie94] for more details on the categorical interpretation of LL.

4.3.4 Lax typing

When embarking in the proof of the analog of Theorem 4.8 for DiLL, one unfortunately stumbles on a counterexample if truly pure nets are employed. This is shown in Figure 4.12: the subnet on the right is a known diverging switching acyclic pure net (it is directly derived from $(\delta)\delta$ of λ -calculus), with however its cut exposed. If the par fires with the tensor the net “explodes”, and this can be achieved by firing the dereliction. If we however fire the codereliction the

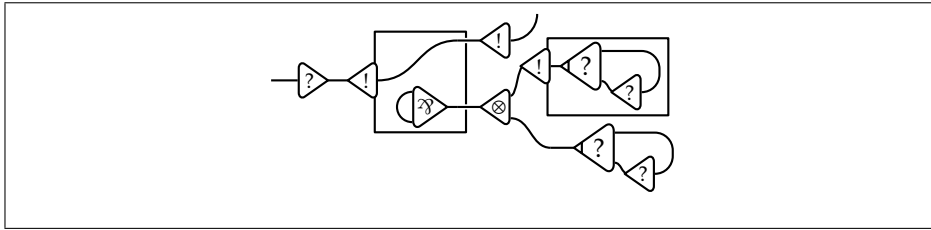


Figure 4.12: A counter example to being weakly normalizable by non erasing steps being equivalent to being strongly normalizing.

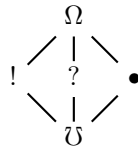
clash on the tensor becomes permanent (unless we erase the box), and the net normalizes by non erasing steps.

We will see in Chapter 7 that this is due to the fact that non erasing reduction is not confluent per se in DiLL, it must rely on erasing steps. This is a striking difference with LL, which in [PTdF08] was shown to have the theorems regardless of these “Lazarus” clashes.

In order to circumvent this problem, we introduce a very weak notion of typing, not to be confused with the *weak typing* presented by Ehrhard and Regnier in [ER06b]. We will therefore call it *lax typing*, as it does not impose a discipline: every net will be typed, and rather it will be used to render exponential clashes permanent.

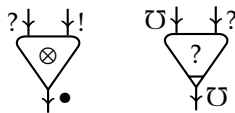
We introduce a set of five types: $?$, $!$, \bullet , Ω and $\bar{\cup}$, with $?^\perp = !$ and all the rest autodual. The type \bullet will stand for a multiplicative type, inferred from \wp and \otimes . We make it autodual because multiplicative clashes do not pose any threat. The types Ω and $\bar{\cup}$ are respectively a sort of polymorphic type indicating that the wire is not really typed yet, while $\bar{\cup}$ will indicate that the wire is, or most importantly was, an exponential clash.

These types are endowed with an order \sqsubseteq , with $?$, $!$ and \bullet incomparable and Ω and $\bar{\cup}$ the top and bottom elements respectively.



It is not really a subtyping order as it is not reversed by duality. However given two such types the type $\alpha \sqcap \beta$ is defined as the meet of the two. So for example $? \sqcap ! = \bar{\cup}$ and $\bullet \sqcap \Omega = \sqcap$.

The typing rules are shown in Figure 4.13. All must be intended *downward closed* with respect to \sqsubseteq , so that for example



are both admissible typings.

Up to now there is no real difference with the usual definition of typing, though we may already see that every net is typable in this way, by just assigning $\bar{\cup}$ to all wires. What changes is the behaviour during glueing and thus reduction.

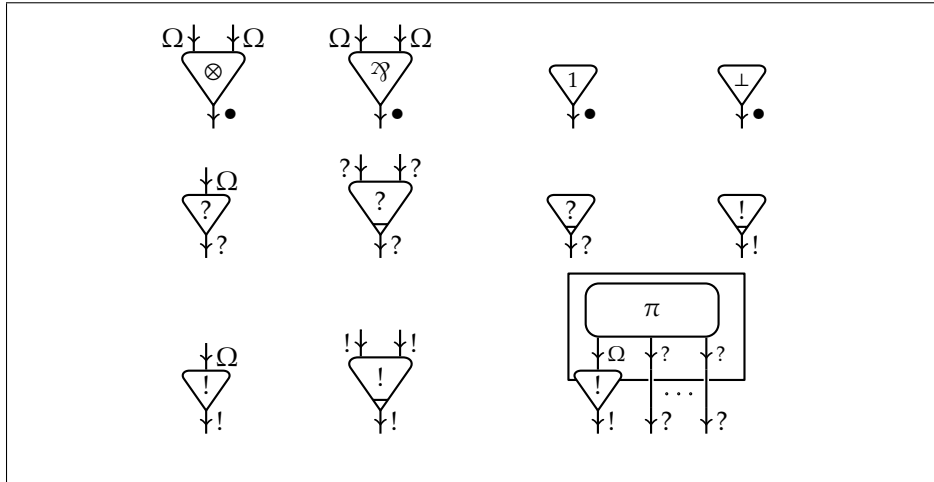


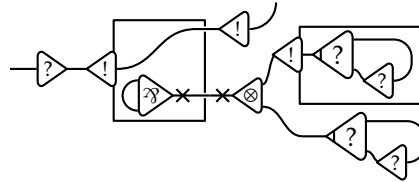
Figure 4.13: Rules for lax typing.

We relax the condition on glueing, permitting it with any type of the internal interface of $\omega[\]$ and the interface of a module μ . However the typing of $\omega[\mu]$ will be obtained from the one of ω and the one of μ by setting for each directed wire e the type $\alpha_1 \sqcap \dots \sqcap \alpha_n$ where α_i are all the types of the directed wires that got melded together during glueing.

This entails that lax typing can change during reduction, and the point is exactly that: we are far from asking that clashes do not occur, that would be too strong a typing in our view. We just want to register when a clash occurs.

In fact we may only mark on the net when a wire is typed by \mathcal{U} (which is independent of the direction of the wire), and we do so by drawing a cross on it. Then, we redefine our reduction to never reduce \mathcal{U} -typed wires. This applies to all cut reduction and also to **n**-redexes, requiring that the wire between (co)contraction and (co)weakening be exponentially typed⁴. It is important to note that therefore such typing is part of the net to be reduced, and is not simply an inert certificate we can throw away after typing. However it is simple enough to consider such nets with lax typing as the pure nets of DiLL, as we will do in this thesis from now on.

So for example the net in Figure 4.12 gets minimally two marks in the following way:



Now when we open the box with the dereliction, the \mathcal{U} type remains on the wire and the net is seen both to weakly normalize by non erasing steps and strongly normalizing by regular ones.

Notice that whenever a type system is clash-free, it can be safely mapped

⁴There is a counterexample analogous to the one we showed also for **n**-reduction.

to this one without the \mathcal{U} -type ever arising (and thus without changing the reduction).

Part II

Determinism: Hypercoherent Spaces and Linearity

In this part we will develop the issue presented in Chapter 2.3, working on the semantic correctness of MALL with respect to hypercoherent spaces.

We will define a new geometrical criterion, *hypercorrectness* (Definition 5.1), such that

- every hypercorrect saturated MALL net θ is **HCoh**-correct (Theorem 5.3;
- every cut free **HCoh**-correct MALL net θ is hypercorrect (Theorem 5.7.

Moreover we will prove by semantical means that such criterion is stable under reduction (Theorem 5.11). The results of this part were the object of [Tra08a]. However the proofs presented here, especially the one for soundness, are a significant improvement to that work.

Chapter 5

Hypercorrectness

We will now go on defining the geometric counterpart to **HCoh**-correctness. The criterion will be a variation on the one by Hughes and van Glabbeek from [HvG03], much like visible acyclicity is a variation on plain switching acyclicity [Pag06a].

5.1 The Criterion

In this section we will redefine the correctness graph. The main difference with \mathcal{G}^{HvG} (see page 3.2.3), is that rather than draw jumps as high as possible, we draw them as low as possible, on the cells where two simple nets meet. A short discussion on our reasons to adopt such a convention is given after the proof of Theorem 5.3, at page 5.2.

5.1.1 Jumping from Contractions

Let us fix a MALL net A , not necessarily satisfying the resolution condition. Given two simple nets λ and μ in it, we say that $c \notin \&2(\lambda, \mu)$ is an **additive contraction** (or simply contraction here that we speak solely of MALL) if

- either c is in both λ and μ but has an incident edge that is not in one of the two¹,
- or c is a *positive* node (\oplus or \otimes ²) just above a cut in λ which is not shared with μ .

The second case is a technical one: if we want to jump as low as possible, we also need to jump from cuts. The set of all contraction of all pairs $\lambda, \mu \in A$ is denoted by $\text{contr}(A)$. It is clear that a contraction can only be either a leaf with discarding axioms over it, or a binary \oplus .

Let $\mathcal{G}(A)$ (the **correctness graph**) be $G(A)$ with

axioms: an edge added between a and a' for every axiom $\{a, a'\} \in \bigcup_{\lambda \in A} \ell(\lambda)$;

¹We called these *partial contractions* in [Tra08a], but we concentrate on these here.

²Recall that in nets there is no axiom cut, though adding it back in as in [HvG03] poses no problems.

jumps: an edge between c and $w \in \&2(A)$ if there are $\lambda, \mu \in A$ with

- $\&2(\lambda, \mu) = \{w\}$;
- c is a contraction for λ and μ .

So with respect to $\mathcal{G}^{\text{HvG}}(A)$, the difference is that we jump from \oplus s and cuts, and from axioms only if the leaf is in both the simple nets. We identify the axiom edges with the axioms in $\bigcup_{\lambda \in A} \ell(\lambda)$.

Switching paths are defined just like in \mathcal{G}^{HvG} : pars and binary withs are switching, and jumps are premises to their $\&$.

In $\mathcal{G}(A)$ we call **total** every node and edge in $\bigcap G(A)$ and every axiom in $\bigcap_{\lambda \in A} \ell(\lambda)$, otherwise it is said to be **partial**. In particular all jumps are partial.

5.1.2 $\&$ -oriented and Compatible Paths

The equivalent restating of the toggling condition given in Proposition 3.10 states that every union of switching cycles has a binary with outside it “justifying” it. We will now prune the unions of cycles to be considered, by both restricting the paths and when they can be united.

First, we consider oriented paths rather than unoriented ones. The two restrictions we impose, *$\&$ -orientedness* of each path and *compatibility* between them, rely in fact on direction. They may be summarized by the following picture, showing what we do *not* allow for the union of cycles we consider.



Formally, we say that an oriented switching path ϕ in $\mathcal{G}(A)$ is $\&$ -oriented if every binary $\&$ in it is traversed downward, i.e. if $w \in \&2(A)$ and $w \in \phi$ and $(w, x) \in \phi$ if and only if (w, x) comes from the principal port of w . In particular jumps in ϕ are always oriented towards their $\&$.

We furtherly say that two oriented paths ϕ and ψ are **compatible** if every edge in both of them are oriented the same, i.e. if $d \in \phi$ then its reversal $d^\perp \notin \psi$. A union S of oriented paths is said to be **compatible** if its cycles are two by two compatible.

Definition 5.1 (Hypercorrectness). A MALL net θ is said to be **hypercorrect** if for every $A \subseteq \theta$ and every non empty *compatible* union S of $\&$ -oriented cycles in $\mathcal{G}(A)$, there is $w \in \&2(A)$ with $w \notin S$.

The idea that oriented paths describe in a better way what is visible to semantics in general is not new. Visible paths are oriented, and in MALL the results in both [AM99] and [BHS05] rely on orienting paths in Girard’s additive proof structure just like that. Also in [MM08] there is a similar idea in the context of game semantics and MLL.

We will also use an even stronger restriction of paths which we call **strictly $\&$ -oriented**. These are $\&$ -oriented ones that furtherly have for each $d \in \phi$ partial:

- if d is not an axiom or cut, then $d = \downarrow d$ is downward;

- if d is a cut then it is oriented towards a positive node, \oplus , \otimes or non negated variable;
- if d is an axiom, it is oriented towards the positive leaf;
- if d is a jump from a contraction c , then c is either the positive node of a cut, or a total contraction.

The last condition follows the intuition of really going as low as possible in the net before jumping, and can be enforced by deleting other jumps from $\mathcal{G}(\Lambda)$. The condition on axioms is to forbid paths going up and down axioms and cuts.

We will call **weak hypercorrectness** the usual hypercorrectness condition where strict $\&$ -orientedness takes the place of the ordinary one. It is indeed an a priori weaker condition, however we will prove in Proposition 5.10 by semantical means (after we prove both sides of the equivalence) that the two are equivalent. Lax hypercorrectness will suffice for the analog of Theorem 3.16, i.e. it implies **HCoh**-correctness.

Weak hyper correctness also gives us the following implication.

Proposition 5.2. *The toggling condition implies weak hypercorrectness.*

Proof. We transform any strictly $\&$ -oriented cycle ϕ in $\mathcal{G}(\Lambda)$ into a switching cycle in $\mathcal{G}^{\text{HvG}}(\Lambda)$ containing the same $\&$ s. In particular, take any jump $j \in \phi$ from a contraction c to a with w . Suppose the jump is justified by simple nets λ and μ . If c is a leaf then j is in $\mathcal{G}^{\text{HvG}}(\Lambda)$ also. If it is a total binary plus there are leaves a and b above c so that $a \in \lambda \setminus \mu$ and $b \in \mu \setminus \lambda$, so that both must have a jump to w in $\mathcal{G}^{\text{HvG}}(\Lambda)$. If finally it is the positive node of a partial cut, then there must be a above the left of the cut and b above the right with $a, b \in \lambda \setminus \mu$, again both with jumps. Now let ψ_c^a and ψ_c^b be the two straight paths in $G(\Lambda)$ from c to a and b respectively, with appended their respective jumps to w . Such paths depart from c in different directions.

One of the two must not intersect ϕ , as otherwise by strict $\&$ -orientedness ϕ would have three incident edges of c : once ψ_c^a and ψ_c^b touch ϕ their initial segments must both be in ϕ (as it must go as low as possible), and j would be the third edge. Let therefore ψ_j be such path in these two cases (total \oplus and partial cut), and j itself when c is a leaf.

For any jumps i and j , ψ_i and ψ_j cannot intersect, as they can do so only if the contraction of i is nested over the one of j or viceversa, which is forbidden by the third clause of strict $\&$ -orientedness. Therefore, substituting in ϕ all jumps j with ψ_j gives a cycle in $\mathcal{G}^{\text{HvG}}(\Lambda)$, which clearly traverses at least the same binary $\&$ s. \square

Without strictness of $\&$ -orientedness it would be much more difficult to prove the above proposition in such a direct way, though such implication is true by Proposition 5.10.

Two examples. Revisiting the examples shown in Figures 3.9 and 3.10, we show in Figures 5.1(a) and 5.1(b) respectively one of their correctness graphs.

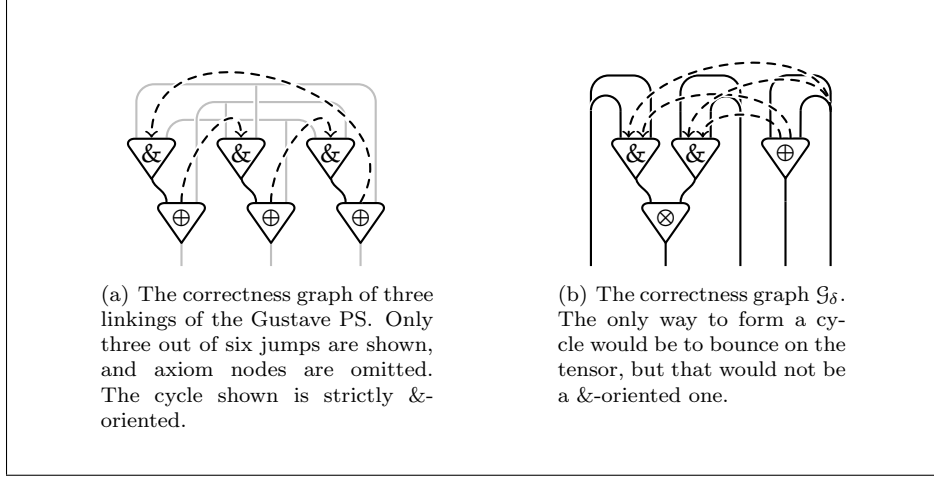


Figure 5.1: Two examples of our version of correctness graphs. The first one shows the rejection of the Gustave net by the criterion, while the second structure, is hypercorrect.

5.2 Hypercorrectness Implies **HCoh**-correctness

Theorem 5.3. *A saturated weakly hypercorrect MALL net is **HCoh**-correct.*

The rest of this section is dedicated to prove the above result. Notice the saturation condition, very similar to the resolution one. This theorem together with Proposition 5.2 gives the first proof of Theorem 3.19 entirely based on switching paths.

Before going into the actual proof, we need some lemmas.

We here clearly make use of the properties S1–3 described at page 55. For reference we write them again here. If Λ is saturated then

(S1) Λ^w is saturated;

(S2) for every $\lambda \in \Lambda$ there exists a $\lambda_w \in \Lambda^w$ with $\lambda \stackrel{w}{\sim} \lambda_w$;

(S3) for every $\lambda, \mu \in \Lambda$, if $\lambda \stackrel{x}{\sim} \mu$ then $\lambda_w \stackrel{x}{\sim} \mu_w$ for some $\lambda_w, \mu_w \in \Lambda^w$ with $\lambda_w \stackrel{w}{\sim} \lambda$ and $\mu_w \stackrel{w}{\sim} \mu$.

Let a **bottom contraction** be one from which jumps are admissible in strictly $\&$ -oriented paths, i.e. either a total one or a positive node above a partial cut. We write $c \rightsquigarrow w$ to mean that there is a jump from c to w .

Lemma 5.4. *For Λ saturated, $w \in \&2(\Lambda)$, c a bottom contraction, e any edge of c in $\mathcal{G}(\Lambda)$, if $c \in \mathcal{G}(\Lambda)$ but $e \notin \mathcal{G}(\Lambda^w)$ then $c \rightsquigarrow w$ in \mathcal{G}_Λ .*

Proof. This is the version in our setting of Hughes and van Glabbeek’s Lemma 4.31 in [HvG03].

Let us first settle the case in which e is not a jump. It must be an edge above c , or c would not persist in $\mathcal{G}(\Lambda^w)$. e must be in a simple net λ which is not in Λ^w , so let us fix it, and let us take $\lambda_w \in \Lambda^w$ from property S2, so that necessarily $\&2(\lambda, \lambda_w) = \{w\}$. Let us inspect the three cases.

- If c is the positive node of a cut d , then $d \notin \lambda$, so λ and λ_w justify $c \rightsquigarrow w$.
- If c is a total leaf, then e is an axiom in $\ell(\lambda)$. By totality also λ_w has an axiom over it, and it cannot be e , so again the jump $c \rightsquigarrow w$ is in $\mathcal{G}(\Lambda)$.
- If c is a total \oplus then λ_w must pick an edge above it, and it does not pick e , so c is binary in $G(\lambda, \lambda_w)$ and we have the jump $c \rightsquigarrow w$.

Now suppose e is a jump $c \rightsquigarrow x$. There are $\lambda \stackrel{x}{\neq} \mu$ with $c \in \text{contr}(\lambda, \mu)$, and if we take λ_w and μ_w we know by **S3** that $\&2(\lambda_w, \mu_w) \subseteq \{x\}$. Now, as e is not in \mathcal{G}_{Λ^w} , c cannot be a contraction for either of the pairs λ, λ_w and μ, μ_w .

- if c is the positive node of a partial cut, then the latter means that $c \in \lambda$ iff $c \in \lambda_w$, and the same for μ and μ_w . Therefore $c \in \text{contr}(\lambda_w, \mu_w)$.
- if c is total, then the above implies that λ and λ_w (and similarly μ and μ_w) have the same edges above c , so again $c \in \text{contr}(\lambda_w, \mu_w)$.

In any case, $c \rightsquigarrow e$ is in $\mathcal{G}(\Lambda)$. \square

Lemma 5.5. *For Λ saturated, every c bottom contraction has a jump $c \rightsquigarrow w$ in \mathcal{G}_{Λ} .*

Proof. For a positive node of a partial cut this is trivial by definition. For a total contraction, take a minimal saturated $\Lambda_0 \subseteq \Lambda$ such that $c \in \text{contr}(\Lambda_0)$. Take any $w \in \&2(\Lambda)$ (it cannot be empty) and consider Λ_0^w , which is saturated by **S1**. By hypothesis $c \notin \text{contr}(\Lambda_0)$, but $c \in \mathcal{G}(\Lambda_0^w)$ by totality, so we are in the hypotheses of Lemma 5.4, and $c \rightsquigarrow w$ in $\mathcal{G}(\Lambda_0) \subseteq \mathcal{G}(\Lambda)$. \square

Lemma 5.6. *If Λ is saturated and weakly hypercorrect, then every non empty compatible union S of strictly $\&$ -oriented cycles has a strictly $\&$ -oriented jump out of it, i.e. $\exists w \in \&2(\Lambda) \setminus S$ and $c \in S$ bottom contraction such that $c \rightsquigarrow w \in \mathcal{G}_{\Lambda}$.*

Proof. This is our equivalent of Lemma 4.32 in [HvG03].

Take a minimal saturated $\Lambda_0 \subseteq \Lambda$ with S still in $\mathcal{G}(\Lambda_0)$. By weak hypercorrectness, there is $w \in \&2(\Lambda_0)$, $w \notin S$. Consider the saturated Λ_0^w : S cannot exist anymore in $\mathcal{G}(\Lambda_0^w)$ by minimality, so there is $e \in S$ such that $e \notin \mathcal{G}(\Lambda_0)$, so e is necessarily partial. Consider the cycle ϕ containing e . Backtracking on ϕ the opposite way with respect to its direction, starting from e , means to go up in the partial part of $\mathcal{G}(\Lambda_0)$ through edges that also are not in $\mathcal{G}(\Lambda_0^w)$, or e would be also. The cases are the following

- We arrive to (or e itself was) an axiom a . By strictness ϕ cannot neither have mounted upward a partial edge, nor traversed a partial cut before going into a . So a must come from a total contraction c , and a and c lie in the hypothesis of Lemma 5.4.
- Otherwise we arrive to (or e itself was) a jump j from a bottom contraction c . Again c and j , which still cannot be in $\mathcal{G}(\Lambda_0^w)$, satisfy the hypothesis of Lemma 5.4, and $c \rightsquigarrow w$.

The jump is in $\mathcal{G}(\Lambda_0)$ and therefore also in $\mathcal{G}(\Lambda)$. \square

We are now ready to prove the main theorem of the section. The following is a much simpler proof than the one we gave in [Tra08a], which gave an actual algorithm for a trip in $\mathcal{G}(\Lambda)$, involving a termination proof.

Proof of Theorem 5.3

Let us fix an interpretation $\llbracket \cdot \rrbracket$ and experiments $\mathbf{e}_1, \dots, \mathbf{e}_k$ in Λ . Without loss of generality, we may suppose that Λ is minimal with respect to being saturated and containing all the linkings $\lambda_1, \dots, \lambda_k$ on which the experiments are taken. By minimality, $\&2(\Lambda) = \&2(\lambda_1, \dots, \lambda_k)$. Also, for every total edge e of $\mathcal{G}(\Lambda)$, all the experiments are defined. If furtherly e is oriented, it makes sense to speak about the hypercoherence of $\{\mathbf{e}_i(e)\}$ in $\llbracket \tau(e) \rrbracket$.

As we did in the proof of Theorem 3.16, we first build a directed graph \mathcal{H} out of $\mathcal{G}(\Lambda)$, following these steps:

- delete all total edges e such that $\llbracket \mathbf{e}_i(e) \rrbracket$ (i.e. it is a singleton), and all orphaned nodes thereafter;
- for each total \mathfrak{A} with auxiliary ports p and q such that $\llbracket \mathbf{e}_i(p) \rrbracket$ and $\llbracket \mathbf{e}_i(q) \rrbracket$, turn q into a separate node disconnected from the \mathfrak{A} ;
- orient all remaining total edges e so that $\llbracket \mathbf{e}_i \rrbracket$;
- orient all partial edges that are not axioms or cuts downward;
- orient all partial cuts and axioms towards their positive node.

Contrarily to MELL, \mathcal{H} need not be acyclic. But we have the following properties:

- each directed elementary path in \mathcal{H} is strictly $\&$ -oriented;
- every two paths in \mathcal{H} are compatible.

The second point is immediate. For the first, take an elementary path ϕ in \mathcal{H} . ϕ cannot bounce on \mathfrak{A} s because we explicitly forbid it with step 2. It also cannot bounce on a $\&$ because all the edges above a binary $\&$ are necessarily partial and directed downward. Finally, all the edges of $\&$ s are directed according to strict $\&$ -orientedness, as if the edge under a binary $\&$ is total its strict hypercoherence orients it away from the $\&$.

Now, we turn \mathcal{H} into a dag by collapsing all cycles. Let \sim be the relation on nodes of \mathcal{H} defined by

$$x \sim y \iff x \rightarrow^* y \rightarrow^* x,$$

i.e. if there is path from x to y and vice versa. This relation is clearly an equivalence one by transitivity of the transitive closure \rightarrow^* . It is important to note that $x \rightarrow^+ y$ iff there is a non empty elementary path from x to y , also when the two are equal³.

Now take the graph \mathcal{H}/\sim , where the nodes are the equivalence classes $[x]$, and we have edges $[x] \rightarrow [y]$ if $[x] \neq [y]$ and $\exists x' \in [x], y' \in [y]$ with $x' \rightarrow y'$ in \mathcal{H} . \mathcal{H}/\sim is a dag, as if there was

$$[x_0] \rightarrow [x_1] \rightarrow \dots \rightarrow [x_k] \rightarrow [x_0]$$

³This is always true in directed graphs: take the first node z along the path that appears multiple times in it. Then if we concatenate the initial segment from x to z and the final segment from the last occurrence of z to y we get an elementary path.

then we would have

$$x_0 \rightarrow^* x_1 \rightarrow^* \cdots \rightarrow^* x_k \rightarrow^* x_0$$

which would give $x_i \sim x_{i+1}$ for all i , which is explicitly forbidden.

Now, \mathcal{H}/\sim must have a sink, and once again, if we prove that only conclusions p (which necessarily form a singleton equivalence class $\{p\}$ in \mathcal{H}/\sim) can be sinks, we are done, as we would have $\wedge\{\mathbf{e}_i(p)\}$ as soon as \mathcal{H} is not empty, which certainly happens if $\neq\{\mathbf{e}_i(A)\}$.

So let us inspect all the cases for node $[c]$ of \mathcal{H}/\sim . Total multiplicative units have necessarily been deleted by step 1, and the new nodes introduced by step 2 cannot be sinks by construction.

- $[c] = \{c\}$ with c total tensor, giving type $A \otimes B$. All of the edges of c are total, and if the total edge d under c has $\wedge\{\mathbf{e}_i(d)\}$ ($\llbracket A \rrbracket \otimes \llbracket B \rrbracket$) then we are done. If otherwise $\vee\{\mathbf{e}_i(d)\}$ ($\llbracket A \rrbracket \otimes \llbracket B \rrbracket$) (as equality would mean the erasure of c in step) then $\vee\{\mathbf{e}_i(p)\}$ or $\vee\{\mathbf{e}_i(q)\}$ for the two auxiliary ports of c , as the two latter sets are the projections of the former, so we have an outgoing edge from c .
- $[c] = \{c\}$ with c total par, giving type $A \wp B$. As above, if $\vee\{\mathbf{e}_i(d)\}$ then $\succ\{\mathbf{e}_i(p)\}$ and $\succ\{\mathbf{e}_i(q)\}$, (so in particular neither of the two is detached by step 2) and one of the two must be strict.
- $[c] = \{c\}$ with c a total unary additive. Both edges above and under it are total, and hypercoherence and equality are preserved between the edge above and under c , so $[c]$ cannot be a sink. The same applies to total leaves that are not contractions.
- $[c] = \{c\}$ with c a total binary with with type $A \& B$. The edge d under it has necessarily $\wedge\{\mathbf{e}_i(d)\}$ ($\llbracket A \rrbracket \& \llbracket B \rrbracket$);
- $[c] = \{c\}$ with c partial and not the positive node of a cut. Then either it is the negative node of a cut, in which case both the cut and the axiom leads away from it, or it has a partial edge under it that does the job.
- $[c] = \{c\}$ with c a bottom contraction. By Lemma 5.5 it has a jump out. This ends the cases for singleton nodes, as total binary \oplus are necessarily bottom contractions.
- $\#[c] \geq 2$. Each node $x \in [c]$ lies in a non empty cycle, as there is $x \neq y \in [c]$ with $x \rightarrow^+ y \rightarrow^+ x$. By the above remarks, we get an elementary non empty C_x with $x \in C_x$ which is strictly $\&$ -oriented. Also for every $y \in C_x$ we have $x \sim y$. Therefore, by looking at nodes only, we clearly have

$$[c] = \bigcup_{x \in [c]} C_x,$$

and such union is compatible. By Lemma 5.6 we get a jump $[c] \ni x \rightsquigarrow w$ with $w \notin [c]$, i.e. $[w] \neq [c]$ so that $[c] \rightarrow [w]$. \square

We briefly explain now why we changed the correctness graph. We have seen that in the proof we need to give an orientation to all partial wires, for which

we do not have a direction given by hypercoherence. As we need the jumps, in $\mathcal{G}^{\text{HvG}}(\Lambda)$ we would be tempted to orient such wires upward to the axioms, but this could break the switching of &s. So one would have to fiddle with different orientation of partial wires. At this point drawing jumps from anywhere in the net could do the trick (such a variation is already proposed in [HvG03], and is what happens in Girard's additive proof nets in [Gir95]), but this would not ease much the way, as we would anyway have to deal with what we call bottom contractions.

5.3 HCoh-correctness Implies Hypercorrectness

Theorem 5.7. *A cut free MALL net which is HCoh-correct is hypercorrect.*

Notice that we do not ask saturation for this result. We want to prove that if there is an illegal union of cycles, then we can build an interpretation and experiments in it that give strictly hyperincoherent results.

The following is the main lemma leading us to the proof of the theorem.

Lemma 5.8. *Let Λ be a MALL net, and ϕ_1, \dots, ϕ_k pairwise compatible &-oriented paths in $\mathcal{G}(\Lambda)$. There exists an interpretation $\llbracket \cdot \rrbracket$ and experiments $\epsilon_1, \dots, \epsilon_n$ such that*

(E1) *for each total directed axiom a in some ϕ_j we have $\wedge\{e_i(a)\} (\llbracket \tau(a) \rrbracket)$; notice that the direction of the axiom is fixed because of compatibility;*

(E2) *for each other total axiom a we have $=\{e_i(a)\}$;*

(E3) *for each total contraction leaf x , $\vee\{e_i(x)\} (\llbracket \tau(c) \rrbracket)$.*

Proof. We will build an ad-hoc hypercoherent space with the web based on some partial axioms of Λ .

Let \mathcal{A} be the graph that has total contractions as nodes and an edge between x and y if the axiom $\{x, y\}$ is in $\bigcup_{\lambda \in \Lambda} \ell(\lambda)$. Such graph is bipartite, as we can divide its nodes into the set \mathcal{A}^+ of nodes typed with a type variable and the one \mathcal{A}^- characterized by negated atoms, and axioms are only between dual types.

Given a contraction leaf x , let $\mathcal{A}(x)$ be the set of edges of \mathcal{A} incident in x , and let E be the set of all edges of \mathcal{A} . It is important that if $x \neq y$ then

$$\mathcal{A}(x) = \mathcal{A}(y) \iff \mathcal{A}(x) = \emptyset \quad \text{or} \quad \mathcal{A}(x) = \{\{x, y\}\}.$$

Suppose in fact that $\#\mathcal{A}(x) \geq 2$, then $\bigcap \mathcal{A}(x) = \{x\}$, so that $\mathcal{A}(x)$ identifies x . If $\#\mathcal{A}(x) = 1$ then necessarily $\mathcal{A}(x) = \{x, y\}$.

Let \mathcal{X} be the hypercoherent space given by

- web $|\mathcal{X}| := E + \{\mathbf{c}, \mathbf{i}, \mathbf{n}\}$ (which stand for coherent, incoherent and neutral);
- hypercoherence, given $s \subseteq_{<\omega}^* |\mathcal{X}|$, $\neq s$, defined by

$$\begin{aligned} \wedge s &:\iff \mathbf{c} \in s \quad \text{or} \\ s &= \mathcal{A}(x) \quad \text{for } x \in \mathcal{A}^-. \end{aligned}$$

Note that $\mathbf{i} \in s, \mathbf{c} \notin s$ implies $\sim s$. Let $\Lambda = \{\lambda_1, \dots, \lambda_k\}$ and take the interpretation that maps all variables to \mathcal{X} . We define experiments $\mathbf{e}_1, \dots, \mathbf{e}_k$, with \mathbf{e}_i on λ_i (taking $\lambda_1 = \lambda_2$ and two different experiments if $\# \Lambda$), with the following laws on every axiom a .

1. If $a = \{x, y\}$ is total and $(x, y) \in \phi_j$ for some j , then if $\tau((x, y)) = \alpha$ (resp. α^\perp) set $e_1(a) := \mathbf{c}$ (resp. \mathbf{i}) and $e_i(a) := \mathbf{n}$ for $i > 1$. Experiments are well defined here because of compatibility.
2. If $a = \{x, y\} \in \lambda_i$ is partial and both x and y are total contraction leaves (therefore $a \in E$), set $e_i(a) := a$.
3. If $a = \{x, y\} \in \lambda_i$ is partial only x is a total contraction, with $\tau(x) = \alpha$ (resp. α^\perp), then if $\mathcal{A}(x) = \emptyset$ and $i = 1$ set $e_1(a) := \mathbf{n}$, else set $e_i(a) := \mathbf{i}$ (resp. \mathbf{c}).
4. In every other case, for $a \in \lambda_i$ set $e_i(a) = \mathbf{n}$.

Now let us prove that these definitions satisfy the requirements.

E1 is a direct consequence of point 1 above, as $\wedge\{\mathbf{c}, \mathbf{n}\}$ ($\llbracket a \rrbracket$) and $\wedge\{\mathbf{i}, \mathbf{n}\}$ ($\llbracket \alpha^\perp \rrbracket$). A total axiom a as in **E2** falls into case 4 of the definition, so $\{e_i(\ell)\} = \{\mathbf{n}\}$.

Now take a total contraction x , with $\tau(x) = \alpha$ (resp. α^\perp). There are two cases. One is that x is not connected to any other contraction leaf (i.e. $\mathcal{A}(x) = \emptyset$), in which case $\{e_i(x)\} = \{\mathbf{i}, \mathbf{n}\}$ (resp. $\{\mathbf{c}, \mathbf{n}\}$) by the special case of point 3, and we have strict hyperincoherence. If $\mathcal{A}(x) \neq \emptyset$ it is easy to see that

$$\mathcal{A}(x) \subseteq \{e_i(f)\} \subseteq \mathcal{A}(x) \cup \{\mathbf{i}\}$$

(resp. $\{\mathbf{c}\}$), where the last point may be included or not depending on $\mathcal{A}(x)$ being all the axioms above x or not. Note that such a point must be included if $\mathcal{A}(x)$ is a singleton, as no contraction leaf can have a single axiom on it by definition. Now if $x \in \mathcal{A}^+$: if $\mathbf{i} \in \{e_i(f)\}$, as $\mathbf{c} \notin \{e_i(f)\}$ we have $\sim\{e_i(f)\}$ ($\llbracket a \rrbracket$), and the same if $\mathbf{i} \notin \{e_i(f)\}$ (i.e. $\{e_i(f)\} = \mathcal{A}(x)$), as the non-singleton $\mathcal{A}(x)$ cannot be equal to any $\mathcal{A}(y)$ for $y \in \mathcal{A}^-$ (and therefore different from x). If the type is α^\perp then we have more directly strict hyperincoherence in $\llbracket \alpha^\perp \rrbracket$ whether $\mathbf{c} \in \{e_i(f)\}$ or $\{e_i(f)\} = \mathcal{A}(x)$ by definition. \square

One could notice that in case of $\# \Lambda = 1$, the above proof becomes the core of Retoré's proof of Theorem 3.18 in [Ret97]. In fact in such a case we get that $P\mathcal{N}\mathcal{X}$ is the coherent space with $|\mathcal{X}| = \{\mathbf{c}, \mathbf{i}, \mathbf{n}\}$, $\mathbf{c} \wedge \mathbf{n}$ and $\mathbf{i} \sim \mathbf{n}$, which is the minimal coherent space needed for Retoré's proof.

We are ready for the proof of the theorem.

Proof of Theorem 5.7

Suppose the net Λ is not hypercorrect. There are then compatible $\&$ -oriented cycles ϕ_1, \dots, ϕ_n ($n \geq 1$) in $\mathcal{G}(\Lambda)$ such that $\&2(\Lambda) \subseteq \bigcup \phi_i$. We apply Lemma 5.8 feeding the ϕ_i s to it, and from properties **E1–3** we deduce the following ones:

- (P1) for every d total edge, if there is an edge d' above d (i.e. with an upward, possibly empty, path from d to d') such that $d' \in \phi_j$ for some j , then $\neq \{e_i(d)\}$, i.e. it is not a singleton;

(P2) for every directed total edge $\downarrow d$ directed downward if $\forall j : \downarrow d \notin \phi_j$, then $\simeq \{e_i(d)\} (\llbracket \tau(d) \rrbracket)$.

If we prove such properties we are done, as for all conclusions p , of Λ if $\{x, q\}$ is the total wire on it then clearly (x, q) is not in any cycle so $\simeq \{e_i(q)\}$ by P2. Notice as a technicality that every conclusion has a downward wire above it in $\mathcal{G}(\Lambda)$ (while it is not strictly true in the interaction net λ). Also, and here is the point where cut freeness kicks in, there must be at least a conclusion which has a cycle passing somewhere above it, so P1 gives in particular strict incoherence.

Let us prove the two properties. For P1, if $d' \in \phi_j$ for a j , one can go up d' following ϕ_j (regardless of its direction) and find the last d'' along ϕ_j which is still above d . If d'' is partial, then there must be either a binary additive or a contraction leaf between d and d'' : in the first case, the resulting experiment cannot be a singleton by construction on additives, and also in the second one, because of property E3. If d'' is total, then there are only three cases possible for it to be maximal. Two of them are that it is a total contraction from which ϕ_j jumps or a binary $\&$ ϕ_j jumps to, and these by the same arguments as above give $\neq \{e_i(d)\}$. Last case is that d'' is the edge of a total axiom, to which property E1 gives a non-singleton that tracked down to d again gives $\neq \{e_i(d)\}$.

We prove P2 by induction on the type of d , and as usual reasoning by cases.

Atomic formula: d is under a total leaf x : either x is a total contraction, and we are settled by property E3 which makes the thesis of P2 always true, or it is under a total axiom $a = \{x, y\}$. Now as $\downarrow d \notin \phi_j$ for all j , necessarily $(y, x) \notin \phi_j$ either. So if $a \notin \phi_j$ for all j by E2 we get a singleton for d , otherwise $(x, y) \in \phi_j$ and we get $\simeq \{e_i(d)\} (\llbracket \tau((x, y)) \rrbracket)$ by E1, so $\simeq \{e_i(d)\} (\llbracket \tau(d) \rrbracket)$.

Par: Suppose the par x above d has above it the downward edges f_0 and f_1 , necessarily total. As no path ϕ_j can bounce on x , $\forall j : \downarrow d \notin \phi_j$ implies the same for f_0 and f_1 . Applying induction hypothesis gives hyperincoherence on both and therefore hyperincoherence on d .

Tensor: suppose x has premises f_0 and f_1 . If the hypothesis $\forall j : \downarrow d \notin \phi_j$ applies for both f_0 and f_1 then inductive hypothesis gives us hyperincoherence on both that implies hyperincoherence on d . Otherwise, suppose that for one of the two, say f_0 , there is h with $\downarrow f_0 \in \phi_h$. Because of the hypothesis on d this path must bounce on the tensor and go up f_1 (implying $\neq \{e_i(f_1)\}$ by P1). By compatibility $\forall i : \downarrow f_1 \notin \phi$, which together by inductive hypothesis gives us $\simeq \{e_i(f_1)\}$ and therefore $\simeq \{e_i(d)\}$.

Unary additive: Straightforward application of inductive hypothesis.

Binary with: By hypothesis such x is in some ϕ_j , and by $\&$ -orientedness $\downarrow d \in \phi_j$, so the hypothesis of P2 never applies.

Binary plus: by definition of hypercoherence, the thesis of P2 is always true. \square

5.4 Complements

We here prove two additional results. The first is that weak hypercorrectness is equivalent to plain one. While this result is trivial for cut free nets by applying

the two main theorems, it is not so in the presence of cuts.

The other very important for the future study of the computational contents of such criterion, is the proof of stability under shared cut reduction.

Both results are achieved by semantics means, employing the equivalence proved in the previous sections. In order to do so we need to statically describe nets with cuts using cut free ones. The tool used for this is *exposing* cuts.

5.4.1 Cut Exposure

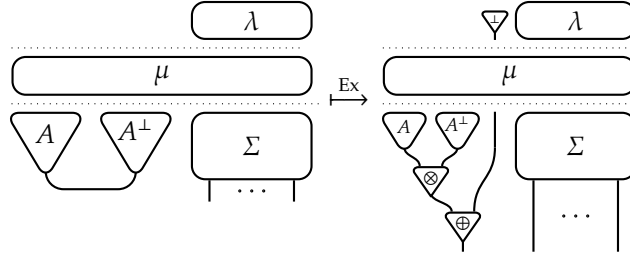
Given a cut formula $A * A^\perp$, with A positive, then its **exposure** is $\text{Ex}(A * A^\perp) := (A \otimes A^\perp) \oplus \perp$, and is extended to cut sequents by

$$\text{Ex}([C_1, \dots, C_k] \Gamma) := \text{Ex}(C_1), \dots, \text{Ex}(C_k), \Gamma.$$

Clearly the leaves of Σ are contained in $\text{Ex}(\Sigma)$. Finally, the exposure of a simple net λ on a cut sequent Σ is the simple net $\text{Ex}(\lambda)$ on $\text{Ex}(\Sigma)$ defined by adding to $\ell(\lambda)$ the \perp of $\text{Ex}(C)$ for each C cut formula in Σ not selected by λ . Clearly

$$\text{Ex}(A) := \{ \text{Ex}(\lambda) \mid \lambda \in A \}.$$

As an informal graphical example, here is what happens when exposing a cut of a slice net with two linkings and one shared cut selected by only one linking.



Notice (though we do not use it here) that cutting all exposed cut formulas in $\text{Ex}(A)$ against the trivial MALL (strongly) correct net having $(A^\perp \wp A) \& 1$ as sequent reduces to A . Trivially A is saturated if and only if $\text{Ex}(A)$ is.

Lemma 5.9. *A is (weakly) hypercorrect if and only if $\text{Ex}(A)$ is (weakly) hypercorrect.*

Proof. Take a (strictly) $\&$ -oriented path ϕ in $\mathcal{G}(A)$, and for each shared cut $C = A * A^\perp$ in Σ , let ϕ_C be the minimal portion of ϕ that touches the roots of A and A^\perp . ϕ_C is a path, i.e. ϕ cannot touch the cut two separate times, because it should bounce on both roots which is impossible as one is switching. We give a path $\text{Ex}(\phi_C)$ (empty if ϕ_C is empty) with the same extremities, all inside the formula substituted for C , and traversing the same $\&$ s. So if $\text{Ex}(\phi)$ is defined by substituting $\text{Ex}(\phi_C)$ for ϕ_C for every shared cut C , yields a (strictly) $\&$ -oriented path in $\mathcal{G}(\text{Ex}(A))$. If in the end we prove that there is a similar mapping in the inverse direction for both strictly and regular $\&$ -oriented paths, and that ϕ is compatible with ψ if and only if $\text{Ex}(\phi)$ and $\text{Ex}(\psi)$ are, we are done.

Let c be the positive node of the cut, d be the cut wire, t and p the \otimes and \oplus added by exposure. We have the following cases.

- ϕ_C bounces on c without traversing any jump: then $\text{Ex}(\phi_C) = \phi_C$.
- ϕ_C bounces on c using a jump j $c \rightsquigarrow w$: if j is also in $\mathcal{G}(A)$ (and in case we are considering weak hypercorrectness if additionally c is still a bottom contraction), then $\text{Ex}(\phi_C) = \phi_C$. Otherwise the jump was necessarily justified by d being a partial cut, which is equivalent to p being a binary plus with a jump j' from p to w in $\mathcal{G}(\text{Ex}(A))$. Substitute j in ϕ_C with the path from c to p with j' .
- ϕ_C traverses d without later traversing any jump from c : substitute d in ϕ_C with a bounce on t .
- ϕ_C traverses d and then a jump j from c to w : as above, if j is also in $\mathcal{G}(A)$ (and is admissible if we are using strictness) then leave ϕ_C be, otherwise descend from the negative node of d down to p and jump to w .

(Strict) $\&$ -orientedness is clearly preserved. Also if ϕ_C and ψ_C are compatible, then so are their replacements. Finally, the above mapping is easily seen to be almost surjective, the only case not covered being when the jumps $c \rightsquigarrow w$ and $p \rightsquigarrow w$ both exist in $\mathcal{G}(\text{Ex}(A))$, and a path ψ jumps from p . In such a case we can safely map it to one using $c \rightsquigarrow w$ in $\mathcal{G}(A)$, bouncing on c if ψ came from c , traversing d if ψ came from the negative node. \square

We immediately get the equivalence between the two forms of the criterion (in case A is saturated).

Proposition 5.10. *If A is saturated, its weak hypercorrectness and hypercorrectness are equivalent.*

Proof. One implication is trivial. For the other, if A is weakly hypercorrect then $\text{Ex}(A)$ is also. By Theorem 5.3, $\text{Ex}(A)$ is **HCoh**-correct, and as it is cut free, by Theorem 3.18 it is hypercorrect. Therefore also A is. \square

One should be aware of the fact that there surely is a way to prove the above by transforming cycles (just apply some cut elimination steps to the chain of implications of the proof!), but it appears to be a very difficult task, while the semantic way is quite easy.

5.4.2 Stability Under Reduction

Theorem 5.11. *If A is a saturated hypercorrect MALL net and $A \rightarrow A'$, then A' is saturated and hypercorrect.*

This section is dedicated to proving that the syntactic hypercorrectness criterion is sound from the point of view of reduction, and is not only an ad-hoc criterion to give equivalence on cut free proofs.

If one takes a look at the proof of stability under reduction of Hughes and van Glabbeek's correctness criterion [HvG03, Section 5.4] one could be surprised by its complexity, especially comparing to the proof in MLL or even MELL. The impression is that part of the sequentialization theorem is reproved. Here, while a similar proof based on paths can be provided, we will give a proof that is in fact completely based on the semantical equivalence. This will give a nice local

proof of stability (while the one on paths must rely on seeing what happens on $\&$ far from the redex and reduct).

First, let us see fix the part about saturation.

Lemma 5.12. *If A is saturated and hypercorrect, and $A \rightarrow A'$, then A' is saturated.*

Proof. The proof is practically identical to the one of [HvG03, Lemma 5.10]. Reducing an axiom or multiplicative cut does not change the $\&$ -resolutions and their relation with linkings. The same happens on an additive cut $A \oplus B * A^\perp \& B^\perp$, with p the plus and w the with, if w is unary in A . Suppose therefore that $w \in \&2(A)$.

If there are in A' two linkings λ and μ with $\&2(\lambda, \mu) = \emptyset$, then in A necessarily $\&2(\lambda, \mu) = \{w\}$ by $\&$ -compatibility. As both survived in the reduction, they must choose different premises of p , so the jump $p \rightsquigarrow w$ is in $\mathcal{G}(\lambda, \mu)$ and forms with the cut a $\&$ -oriented cycle containing the only binary with.

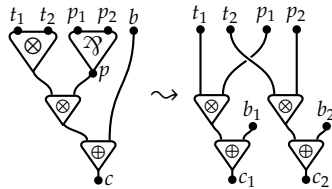
Now for the (pseudo) $\&$ -fullness part. Now suppose there is W $\&$ -resolution with $\&2(W, A') = \&2(A')$ with no corresponding linking. This induces two $\&$ -resolutions W_1 and W_2 for A , given by making the same choices of W and then left and right respectively for w . We have $\&2(W_1, A) = \&2(W_2, A) = \&2(A)$ as $\&2(A') \subseteq \&2(A)$ and $w \in \&2(A)$. So we get $\lambda, \mu \in A$ with $G(\lambda) \subseteq W_1$ and $G(\mu) \subseteq W_2$. Now neither of the two can have survived in the reduction step, or they would be on W . So they must choose on p in the opposite way as they do on w (and must therefore select the cut), and we end up again with a trivial illegal cycle in $\mathcal{G}(\lambda, \mu)$. \square

Proof of Theorem 5.11

First we can exclude the case of the multiplicative unit and axiom cut reductions, as clearly all paths in $\mathcal{G}(A')$ correspond to paths in $\mathcal{G}(A)$ with the same $\&$ s.

For the other cases, we show that if we suppose that $\text{Ex}(A')$ is not **HCoh**-correct, and $\text{Ex}(A)$ is, we arrive at a contradiction. In such a case we can conclude combining Lemma 5.9 and the two main Theorems 5.3 and 5.7. For the purpose of this proof if a cut formula C gets erased during the reduction step because no remaining linking selects it, we can suppose it is still exposed in $\text{Ex}(A')$ by making all linkings choose \perp on $\text{Ex}(C)$. This is just not to make special cases for when the cut is erased. Let Γ and Γ' the exposed sequents of the two nets, which differ only by the reduced exposed cuts.

Multiplicative reduction. $C = A \otimes B * A^\perp \wp B^\perp$. The following is the local picture of the exposure of the redex and of the reduct, without drawing the trees above the cut.

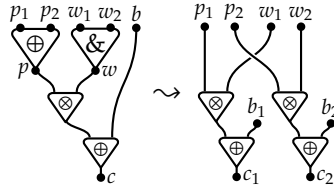


Suppose we have experiments ϵ_i on linkings $\lambda_i \in \text{Ex}(A')$ with $\sim\{\epsilon_i(A')\}$ ($\llbracket \Gamma' \rrbracket$) for a given interpretation $\llbracket \cdot \rrbracket$. These experiments can also be viewed as being on

$\text{Ex}(\Lambda)$ as all axioms in Λ' are also in Λ . Also the results on every other conclusion different from the ones of the involved exposed cuts must coincide, and the \oplus s are whether all binary or all unary (depending on whether the linkings choose or not the ports t_i and p_j). In particular the result on Λ cannot be a singleton either.

By **HCoh**-correctness of Λ we get $\neg\{\epsilon_i(\Lambda)\}$, and as we have \asymp on all free ports different from c , we necessarily have $\neg\{\epsilon_i(c)\}$. Therefore all the \oplus s are unary (i.e. the exposed cut was total). Now chasing up the hypercoherence we get \supset on both t_1, t_2 and p (the latter in $\llbracket A \rrbracket \wp \llbracket B \rrbracket$), and at least one is not a singleton. If we have $=$ on p then the strict hypercoherence of a t_j is inherited by c_j in Λ' . Otherwise we have $\neg\{\epsilon_i(p_j)\}$ which again goes down to c_j .

Additive reduction. $C = A \oplus B * A^\perp \& B^\perp$.



As above from $\neg\{\epsilon_i(\Lambda')\}$ we get $\neg\{\epsilon_i(c)\}$ (again inequality lifts from Λ' to Λ). Chasing up hypercoherence we get \supset on p and w , which implies particularly that the top \oplus in Λ is not binary for λ_i . As the λ_i s are all linkings that survived during the reduction step, also the $\&$ must be unary and choose the same side. We thus get \supset on p_j and w_j for a j , with at least one inequality. This gives us strict hypercoherence on c_j , unless b_j is selected by ϵ_i , but this cannot happen as they would have to select b in Λ . \square

Part III

Nondeterminism: Differential Operators and Resources

Chapter 6

Church-Rosser Theorem for Pure DiLL

This chapter is devoted to proving confluence of pure DiLL° nets. Being in the untyped case means that this result is extremely general, but also that we cannot rely on lemmas based on normalization, such as the modulo version of the Newman’s lemma, Lemma 4.2.

We thus obtain it by proving two results on the so called *developments*, i.e. the reductions of all the redexes present in a term. In λ -calculus such results, a full proof of which can be found on Schroer’s thesis [Sch65], states that if we take a pure λ -term t , we can reduce *all* the redexes of t with freely chosen single reduction steps, obtaining a unique result.

More precisely, reducing only the residuals of redexes of t , leaving alone the redexes created along, is strongly normalizing and confluent. This is not at all trivial, due to the possibility that redexes of t be duplicated.

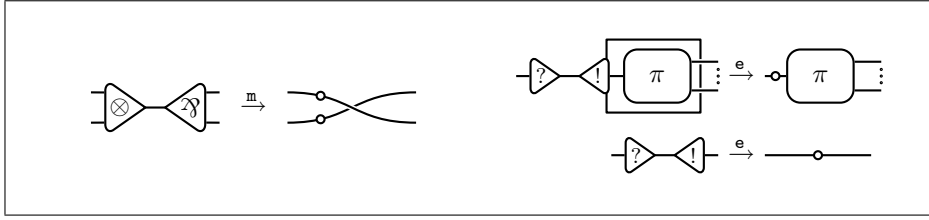
As the reduction of proof nets is more atomic than the one of λ -calculus, the definition is not transferred verbatim. In the next section we will define what are to be considered *new* redexes, we will then state the result and infer confluence from it.

6.1 Marking New Cuts: DiLL°

In [Dan90], Danos proved the counterpart of the finite development theorem for MELL, and Pagani and Tortora de Falco did the same for the whole of second order LL in [PTdF08]. In this setting the actual technical definition of development takes another form, that however exactly correspond to the one of λ -calculus when translating it into nets.

The idea is that only the reductions that in a typed setting would decrease the logical complexity of the cut formula are those to be considered creating actual new redexes. In MELL this means the multiplicative reduction and the dereliction against box rule, while all the other exponential reduction are to be considered creating “old” cuts.

We thus define a notion of new and old cuts, by marking with a circle the wires created by rules that in typed nets would reduce the logic complexity (\otimes on \mathfrak{A} , dereliction on box, dereliction on codereliction), see Figure 6.1), and by

Figure 6.1: The marking rules of DiLL° .

restricting the reduction on unmarked wires only. The effect is the same as for clash wires typed by \mathcal{U} with a lax typing discipline, however we distinguish the two as we need the lock on new cuts to be temporary, while the lock on clash wires must be permanent. We will call DiLL° the system thus obtained. From now on, let \sim and \mathfrak{c} be either $\overset{\mathfrak{a}}{\sim}$ and $\overset{\mathfrak{n}}{\mathfrak{c}}$, or $\overset{\text{asp}}{\sim}$ and $\overset{\text{nzp}}{\mathfrak{c}}$, or finally $\overset{\text{asp}}{\sim}$ and $\overset{\text{nzp}}{\mathfrak{c}}$. See Figure 4.10 at page 87, and Figure 4.3 For their definition.

In fact the multiplicative reduction does not pose many problems, and the great part of the proof is taken up dealing with the exponential reduction. The proof will be totally combinatorial.

Theorem 6.1 (Finite and unique developments). *Reduction on DiLL° pure proof nets is strongly normalizing modulo \sim and Church-Rosser modulo \sim . The reductions $\overset{\text{ec}}{\mathfrak{c}}$ and $\overset{\mathfrak{m}}{\mathfrak{c}}$ are also $\text{CR}\sim$ on their own.*

We naturally split the proof of the theorem in two parts, the strong normalization and the confluence ones. The remainder of the chapter will be thus devoted to the proof of the two parts, with Propositions 6.8 and 6.9 being such final results. For now, we show how it implies its most important corollary: the Church-Rosser modulo property for the whole of pure DiLL.

Theorem 6.2 (Church-Rosser). *Reduction on DiLL pure proof nets is $\text{CR}\sim$, and so are \mathfrak{m} and ec taken alone.*

Proof. Finite developments make it really easy to define a parallel reduction to infer confluence. Clearly DiLL proof nets can be embedded in DiLL° , and DiLL° can be surjected on DiLL. Let here be ψ the latter surjection, that just deletes all the markings. Now in DiLL define $\pi \leftrightarrow \sigma$ if and only if $\pi \overset{\text{mec}^*}{\rightarrow} \sigma'$ in DiLL° and $\sigma = \psi(\sigma')$. Fundamentally, \leftrightarrow reduces any number of redexes, but all must be already present at the start, exactly the way parallel reduction does. Now we have that

- $\overset{\text{mec}}{\mathfrak{c}} \subseteq \leftrightarrow \subseteq \overset{\text{mec}^*}{\mathfrak{c}}$, so that $\leftrightarrow^* = \overset{\text{mec}^*}{\mathfrak{c}}$;
- \leftrightarrow is strongly $\text{CR}\sim$ (see page 70): this is because $\overset{\text{mec}}{\mathfrak{c}}$ is $\text{CR}\sim$ in DiLL° , so that $\overset{\text{mec}^*}{\mathfrak{c}}$ is strongly so.

Then we conclude, as \leftrightarrow is $\text{CR}\sim$ by Lemma 4.4 (\leftrightarrow is reflexive), which means that $\leftrightarrow^* = \overset{\text{mec}^*}{\mathfrak{c}}$ is strongly $\text{CR}\sim$, which in turn by Lemma 4.3 gives Church-Rosser modulo \sim for the ordinary reduction¹.

It is not hard to give parallel reductions for the ec and \mathfrak{m} ones and do the same. \square

¹Notice that we cannot infer $\text{CR}\sim$ of \leftrightarrow directly from the same property in DiLL° , as chained parallel reductions are not necessarily in DiLL° .

Another interesting direct corollary of the developments theorem is the following.

Corollary 6.3. *The **ec**-reduction is strongly normalizing and Church-Rosser modulo \sim on differential λ -nets.*

Proof. Because of the typing discipline the wires that in the **e**-reduction of DiLL° would be marked as multiplicative, and as such cannot be reduced by **e**. Therefore every chain of **ec**-reductions of differential λ -nets is an **ec**-reduction in DiLL° . \square

6.2 Finiteness of Developments

The most technical point of the proof is defining a measure which strictly decreases on **e**-reductions. The other reductions will be far easier to handle, as will be seen in the proof of Lemma 6.7. We will now consider the full version of \sim and **c**, i.e. \sim will be $\overset{\text{asp}}{\sim}$ and **c** will be **nzp**, as any normalization result on those transfers to all subequivalences and subreductions.

6.2.1 Measuring Exponential Reduction

Ideally, we may regard exponential reduction as a procedure that “slides” cells along exponential straight paths in the net, with **!** and **?** cells sliding in opposite direction.

We thus try to assign to each cut a natural number, indicating how far the two cells around it are from the end of the path they are sliding on. After a cut is fired, the cuts created by the reduction would have a lesser weight, but there may be many of them. So we will employ the multiset of the weights of the cuts and the multiset order. Another problem arises: sums make it so that when a reduction creates addends, there is a sort of global duplication of the net.

In [Tra08b] we settled this by employing multisets of multisets. Here we will however be able to estimate how many addends can sprout during reduction, so we can use this value and count each cut as many times as there can be addends containing it.

Another kind of duplication is the one of the box: this happens both because of contractions and because of the codereliction rule, as they extract a linear copy. We will therefore give an estimate of the number of copies that can be made out of a box.

All these estimates are made by inspecting exponential straight paths, and after defining those we will define the measures we need.

Exponential paths. An **exponential path** is a directed path ϕ such that all wires in it are not \circ -marked and are either all laxly typed with **?** or all with **!**. By typing is is clear that such a path is straight, i.e. one that traverses each cell from the principal port to an auxiliary one or viceversa. Also every c internal to ϕ , i.e. not at its extremities, must either be a box, a contraction or a cocontraction. Clearly ϕ does not contain clashes.

Depending on the common type of their wires, we distinguish accordingly between **!-paths** and **?-paths**. Because of switching acyclicity exponential paths




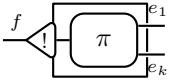
	$?(e) = ?(p)$, the variable associated with the free port p ;
	$?(e) = ?(e_1) + ?(e_2)$;
	$?(e) = ?(f)$;
	$?(e_i) = \begin{cases} ?(f)(1 + \sum_j !(e_j)) & \text{if } \pi = 0, \\ ?(f)(1 + \sum_j !(e_j)) \sum_{\lambda \in \pi} \text{sp}(\lambda) ?(e^\lambda) & \text{otherwise;} \end{cases}$
otherwise:	$?(e) = 1$.

Table 6.1: Rules for the $?(e)$ -weight. $?(e) = 1$ if e is not exponential.

are non repeating, so no loops are possible and the length of paths is bounded. An **exponential wire** is a wire typed with $!/?$ and not \circ -marked.

The measures. We will define by mutual induction three basic measures on which we will base the measure of the whole net. Two of them, the **$?(e)$ -weight** $?(e)$ and the **$!(e)$ -weight** $!(e)$, are on wires. The third, the **spread** $\text{sp}(\lambda)$, is defined on simple nets.

Morally they are natural numbers, however we will need to make the induction work we will need rather to define the dependence of such measures from the environment. Therefore we assign *variables* $?(p)$ (resp. $!(p)$) for each *free* port p laxly typed with $?$ (resp. $!$), and all the measures will be polynomials in such variables. This also ease us when plugging a module in a context, as intuitively we will be able to instantiate such variables.

First we give all the conditions, then we will later prove that they indeed define functions by induction.

Weighting wires and sums. We will define our weights and the spread. First, $?(e) = 1$ if e is not exponential (and in particular if it is \circ -marked); otherwise, let e be oriented so that its type is $!$, and let c be the node in the graph of the simple net to which e points.

In Table 6.1 we provide the laws for $?(e)$, giving them depending on the symbol of c , drawing it on the left of e . By e^λ we denote the wire corresponding to e inside a box, in the net λ of the box contents.

Dually, when e is exponential let e be oriented so that its type is $!$, or otherwise set $!(e) = 1$. Then Table 6.2 provides all the laws for the $!(e)$ -weight.

The spread $\text{sp}(\lambda)$ of a simple net λ is finally defined by the following formula.

$$\text{sp}(\lambda) = \prod_{c \in \varrho_0(\lambda)} !(c) \cdot \prod_{c \in \delta_0(\lambda)} ?(c)$$

where

- $\varrho_0(\lambda)$ (resp. $\delta_0(\pi)$) is the set of all derelictions (resp. coderelictions) at depth 0 in λ ;




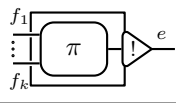
	$!(e) = \begin{cases} !(p) & \text{if } p \in \mathbf{fp}_0(\pi), \\ !(f) & \text{if } p \in \mathbf{fp}_0(\sigma(B)) \text{ for a box } B, p \text{ is above} \\ & \text{an auxiliary port, and } f \text{ is the wire cor-} \\ & \text{responding to } p \text{ outside the box,} \\ 1 & \text{if } p \in \mathbf{fp}_0(\sigma(B)) \text{ for a box } B \text{ and } p \text{ is} \\ & \text{above the principal port;} \end{cases}$
	$!(e) = !(e_1) + !(e_2);$
	$!(e) = !(f);$
	$!(e) = \begin{cases} 1 + \sum_j !(f_j) & \text{if } \pi = 0, \\ (1 + \sum_i !(f_i)) \sum_{\lambda \in \pi} \mathbf{sp}(\lambda) & \text{otherwise.} \end{cases}$
otherwise:	$!(e) = 1.$

Table 6.2: Rules for the !-weight. $!(e) = 1$ if e is not exponential.

- the (! or ?) weight of a cell is the weight of the wire connecting the principal port.

Notice that here is a circular dependency between the three measures, so the next lemma is not trivial.

Lemma 6.4. *Given a DiLL° proof net π , $?(e)$, $!(e)$ and $\mathbf{sp}(\lambda)$ are defined and unique for all $e \in \mathbf{w}_1(\pi)$ and all $\lambda \in \wp_1(\pi)$.*

Proof. Uniqueness is clear. We prove that we can define all of the measures by a nested induction. Suppose we have defined them for all DiLL° proof nets of depth strictly less than $d(\pi)$.

First we show that $!(e)$ is defined for all $\mathbf{w}_1(\pi)$, and at the same time that $?(e)$ and $\mathbf{sp}(\lambda)$ are for $e \in \mathbf{w}_k(\pi)$ and $\lambda \in \wp_k(\pi)$ with $k \geq 1$. This is done by induction on the length of maximal ?-paths at depth 0: starting with e for $e \in \mathbf{w}_0(\pi)$, or with the wire f on the principal port of the box $B \in \mathbf{b}_0(\pi)$ containing e or λ otherwise. In fact all measures inside boxes are already defined by induction hypothesis, but they depend on variables $?(p)$ and $!(p)$ with $p \in \mathbf{fp}_1(\pi)$. However by lax typing (as a variable is assigned only if the port is typed in the “right” way):

- all the variables $?(p)$ upon which the measures depend must be on the ports over a principal port of a box, and we can instantiate them with 1;
- all the variables $!(p)$ are over auxiliary ports, and we can instantiate them with $!(f)$ with $f \in \mathbf{w}_0(\pi)$, defined by secondary induction hypothesis, as these are wires further down ?-paths than the wire on the principal port.

In the other case, $e \in \mathbf{w}_0(\pi)$, $!(e)$ is defined from the !-weight on wires further down ?-paths, and possibly the spread of nets inside a box which have just been seen to be defined, all by secondary induction hypothesis.

Next, $?(e)$ for $e \in w_0(\pi)$ can now be defined by induction on the maximal length of $!$ -paths starting with e , as they depend: on $!$ -weights; on weights $?(f)$ with either $f \in w_1(\pi)$ or further down $!$ -paths; and on $\text{sp}(\lambda)$ with $\lambda \in \wp_1(\pi)$.

Finally, taken any $\lambda \in \wp_0(\pi)$, its spread depends on $!$ and $?$ -weights. \square

Notice that $\text{sp}(\lambda \parallel \mu) = \text{sp}(\lambda) \text{sp}(\mu)$, and most importantly each measure is a non-zero polynomial in $\mathbb{N}[?(p_i), !(q_j)]$ where p_i (resp. q_j) are all the ports in $\text{fp}_0(\pi)$ with type $?$ (resp. $!$). It is also very important that $!$ -weights are polynomials in variables $!(p)$ only: their dependence on $?$ -weights (as they depend on the spreads) does not “exit” boxes. Finally, being polynomials with positive coefficients, all measures are monotonous functions on their variables, and if we take as domain the non zero positive integers \mathbb{N}^* then each measure is extensionally greater or equal than 1.

Weighting nets and polynets. The **weight** $|e|$ of a wire is $?(e) + !(e)$. The weight $|C|$ of a set of wires C is the *multiset* of weight of its elements. Let $!\text{cw}_0(\lambda)$ be the set of exponential cuts at depth 0 of a simple net λ . Let us fix a polynet π . Then for each $\lambda \in \wp_1(\pi)$ define by induction on its depth (or its codepth in π)

$$|\lambda| := |!\text{cw}_0(\lambda)| + \sum_{\substack{B \in \text{b}_0(\lambda) \\ \mu \in \sigma(B)}} \#(B) \text{sp}(\mu) |\mu|,$$

where $\#(B)$, the **count** of the box, is $?(e)(1 + \sum_j !(f_j))$ with e and f_j the wires on the principal and the auxiliary ports respectively.

Finally, let

$$\|\pi\| := \sum_{\lambda \in \pi} \text{sp}(\lambda) |\lambda|.$$

Notice that this measure depends monotonously from the weight of each part of the net. This intuition will be given a solid ground by the modularity Lemma 6.6.

We now try to give an intuitive idea of why the measures are defined in this way. Morally $!(e)$ measures the size of the tree of cocontractions above e (which is invariant under associativity). The most important feature is that it counts all the coderelictions linked to e . On boxes we count

- the $!$ -weight on the auxiliary ports because the codereliction against box rule creates a contraction and a codereliction; plus one to count the box itself, especially if it has no auxiliary ports;
- multiplied by the spread of the contents in order to be invariant by \mathbf{s} -conversion, and keep such invariants even if the sum inside... spreads.

Dually $?(e)$ measures the size of the tree of contractions above e . The rule when e is on an auxiliary port of a box B contains:

- $?(f)$ because the contractions on the principal port of B may shift to auxiliary ports during reduction;
- the sum of $!$ -weights of the auxiliary wires because codereliction against box creates contractions; plus 1 to provide something to decrease when a cut enters a box (box against box and those similar);

- the $?$ -measures inside because either by opening the box or by \mathbf{p} -conversion the contraction trees inside can pour outside; summed to respect both \mathbf{p} -conversion and \mathbf{s} -conversion; however the sum is weighted with the internal spread to avoid that a reduction generating a sum inside could raise such weight.

As already hinted, $\text{sp}(\lambda)$ estimates how many addends may have a reduct of λ . This is achieved by morally multiplying all the possible number of choices potentially to be done in λ . Now sums arise

- on (co)dereliction against co(co)ntraction reductions, so the size of the tree of co(co)ntractions on the principal port of a (co)dereliction should estimate what choices that (co)dereliction may do;
- on (co)dereliction against box rules, when the box contains an actual sum; however the spread of a box contents are already accounted for in both “moral” contraction and cocontraction trees.

Finally, in a simple net we weight cuts with a multiset. However we make the cuts inside a box B count $\#(B)$ times as this number estimates how many regular and linear copies of its contents may be done (by contraction against box and codereliction against box rules respectively), and all cuts count $\text{sp}(\lambda)$ times to account for “additive duplication”.

Replacement and modularity lemmas. The achievements of this part will be some standardized hypotheses to apply to each reduction rule to show that the measure decreases. What in general will be left to check is

- when replacing a redex with a reduct the measures on the interface do not increase, ensuring what is outside does not increase either;
- the spread gets divided among the addends of the reduct, i.e. the sum of the spreads of the reducts is lower than the spread of the redex;
- locally each addend of the redex has less size.

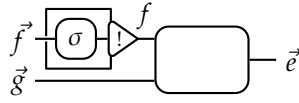
So we will here define when a module can safely replace another (the relation $\mu \preceq \lambda$), prove that this yields the first point in the replacement lemma 6.5, and finally show with the modularity lemma 6.6 that the basic hypotheses on redex and reduct imply that replacing the latter for the former decreases the measure $\|\cdot\|$. Due to the complex nature of the system and of the measures we have defined, this proofs are far from trivial.

In the following, when comparing two measures we employ the extensional (pointwise) order on polynomials $\mathbb{N}[X_i]$. Given a simple module λ , let $?I_\lambda$ and $!I_\lambda$ be the set of the $?$ and $!$ laxly typed ports of the interface I_λ respectively. Define analogously $?I_\omega^\square$ and $!I_\omega^\square$ for a context $\omega[\]$. We extend the weights to free ports, in the sense that if $p \in !I_\lambda$ (resp. $?I_\lambda$) we define $?(p)$ (resp. $!(p)$) to be the measure of the wire above it, the other weight being already defined as a variable.

For different simple modules λ, μ , we distinguish the weights calculated on one or the other by putting them as superscripts, as in $?^\lambda(e)$. If λ and μ have the same interface I (with the same lax typing) we say that μ **can replace** λ (written $\mu \preceq \lambda$) if for all $p \in !I$ and $q \in ?I$ we have $?^\mu(p) \leq ?^\lambda(p)$ and $!^\mu(q) \leq !^\lambda(q)$ respectively.

Lemma 6.5 (replacement). *If $\omega[\]$ is an affine context, $\lambda \preceq \mu$, $\omega[\lambda]$ and $\omega[\mu]$ are proof nets, then for each $e \in \mathbf{w}_1(\omega)$ we have $?^{\omega[\mu]}(e) \leq ?^{\omega[\lambda]}(e)$ and $!^{\omega[\mu]}(e) \leq !^{\omega[\lambda]}(e)$.*

Proof. We can easily reduce to the case of a linear context. We reason by induction on the size of ω : if there is a cell c with a wire on the internal interface, detach it from ω obtaining a smaller context ω' . One must then check that glueing c to both simple nets obtaining λ' and μ' yields still $\mu' \preceq \lambda'$. This really poses no problem because we are adding cells with a single wire attached to the hole for now, and all the measures are defined with monotonous operations. For example (drawing $!I$ left and $?I$ right) in

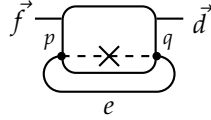


when we check the replacement condition on \vec{e} we get that

$$!^{\mu'}(e) = !^{\mu}(e) \left[(1 + \sum !(\vec{f})) \sum_{\nu \in \sigma} \text{sp}(\nu)!(f) \right]$$

which by the hypothesis $\mu \preceq \lambda$ is less than $!^{\lambda'}$. The other cases for checking $!$ and $?$ weights are analogous. Also checking the spread is straightforward, once the weights have been settled.

The slightly harder part comes if ω has no cells on I_{ω}^{\square} , and we suppose that there is a wire e which connects I_{ω}^{\square} to itself. Supposing that glueing e to λ/μ yields an exponential wire (otherwise it is trivial), the situation can be depicted by the following picture.



There cannot be then any exponential path in neither λ nor μ , as it would form an exponential loop with e . Therefore the variable $!(p)$ (resp. $?(q)$) does not appear in $!^{\lambda}(q)$ and $!^{\mu}(q)$ (resp. $?^{\lambda}(q)$ and $?^{\mu}(q)$). We can therefore safely write

$$\begin{aligned} !^{\mu'}(\vec{d}) &= !^{\mu}(\vec{d}) [!^{\mu}(p)/!(q)] \leq !^{\lambda}(\vec{d}) [!^{\lambda}(p)/!(q)] = !^{\lambda'}(\vec{d}), \\ ?^{\mu'}(\vec{f}) &= ?^{\mu}(\vec{f}) [!^{\mu}(p)/!(q), ?^{\mu}(q)/?(p)] \leq ?^{\lambda}(\vec{f}) [!^{\lambda}(p)/!(q), ?^{\lambda}(q)/?(p)] = ?^{\lambda'}(\vec{f}). \end{aligned}$$

If in ω there are no cells nor wires on the interface, it means that $\omega[\lambda] = \omega' \parallel \lambda$, with ω' the net ω without the wiring on the hole. In any case it is done. \square

Lemma 6.6 (modularity). *Take λ and μ_1, \dots, μ_n with simple modules and $\omega[\]$ a context such that $\omega[\lambda]$ and $\omega[\mu_i]$ are all DiLL° pure proof nets. Suppose the following points are satisfied:*

- for every i we have $\mu_i \preceq \lambda$;
- $\sum_i \text{sp}(\mu_i) \leq \text{sp}(\lambda)$;
- if $n = 0$, then $|\lambda| > []$;

- otherwise, for every i we have $|\mu_i| + |D_i|^{\mu_i} < |\lambda|$ pointwise, where D_i is the set of dormant wires in μ_i that are not dormant in λ .

Then we have the pointwise inequality

$$\|\omega[\sum_i \mu_i]\| < \|\omega[\lambda]\|.$$

Moreover if the inequality in the last point of the hypotheses is replaced by \leq , then so is the above one.

We recall that a dormant wire is one over a free port which is not an axiom, the only kind that can become cut after glueing.

Proof. Let us reason by induction on the depth of the hole in $\omega[]$. Let $\mu = \sum_i \mu_i$.

Base step. If $\omega[] = \delta[] + \pi$ is affine with $\delta[]$ linear, then it suffices to show the thesis for $\delta[]$. If $n = 0$ then trivially

$$\|\delta[\lambda]\| \geq |\lambda| > [] = \|0\| = \|\delta[0]\|.$$

Suppose therefore $n \geq 1$. By the replacement lemma we have that for every i , all the weights of wires in δ are less in $\delta[\mu_i]$ than in $\delta[\lambda]$, so $|\delta|^{\delta[\mu_i]} \leq |\delta|^{\delta[\lambda]}$. If moreover C denotes the set of cuts of $\delta[\lambda]$ that are on the interface, and C_i the same for $\delta[\mu_i]$, then clearly $C_i \subseteq C \cup D_i$, and $|C_i \setminus D_i|^{\delta[\mu_i]} \leq |C|^{\delta[\lambda]}$, as such wires are in δ and the replacement lemma applies. For the same reason when we instantiate the variables we get $|\mu_i|^{\delta[\mu_i]} + |D_i|^{\delta[\mu_i]} < |\lambda|^{\delta[\lambda]}$. We thus have

$$\begin{aligned} |\omega[\mu_i]| &= |\omega|^{\delta[\mu_i]} + |C_i|^{\delta[\mu_i]} + |\mu_i| \\ &\leq |\omega|^{\delta[\lambda]} + |C_i \setminus D_i|^{\delta[\mu_i]} + |D_i|^{\delta[\mu_i]} + |\mu_i| \\ &< |\omega|^{\delta[\lambda]} + |C|^{\delta[\lambda]} + |\lambda|^{\delta[\lambda]} = |\delta[\lambda]|. \end{aligned}$$

As for the spread, we have

$$\sum_i \text{sp}(\delta[\mu_i]) = \text{sp}^{\delta[\mu_i]}(\delta) \sum_i \text{sp}^{\delta[\mu_i]}(\mu_i) \leq \text{sp}^{\delta[\lambda]}(\delta) \cdot \text{sp}^{\delta[\lambda]}(\lambda) = \text{sp}(\delta[\lambda]).$$

So we conclude this step by

$$\|\delta[\sum_i \mu_i]\| = \sum_i \text{sp}(\delta[\mu_i]) |\delta[\mu_i]| < (\sum_i \text{sp}(\delta[\mu_i])) \cdot |\delta[\lambda]| \leq \text{sp}(\delta[\lambda]) |\lambda| = \|\lambda\|.$$

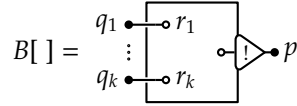
Clearly replacing \leq for $<$ is still valid.

Inductive step. Suppose now that ω is not affine, let $B[] \in \mathfrak{b}_!(\omega)$ be the box containing the hole with maximal depth, so that its contents is an affine context $\delta[] + \pi$, and let $\psi[]$ be the context in ω relative to the subnet B , i.e. $\omega[] = \psi[B[]]$. If $n = 0$ by inspecting the laws of the measures one easily sees that $B[\pi] \preceq B[\delta[\lambda] + \pi]$, $\text{sp}(B[\pi]) = 1 = \text{sp}(B[\delta[\lambda] + \pi])$ and

$$|B[\pi]| = \#(B) \|\pi\| < \#(B)(\|\delta[\lambda]\| + \|\pi\|) = |B[\delta[\lambda] + \pi]|,$$

so we can apply inductive hypothesis and get the result.

If $n \geq 1$, applying the replacement lemma we get that the measures in $\delta[\mu_i]$ are pointwise lower than the ones in $\delta[\lambda]$, and we can instantiate them with the variables of $B[\]$ (and 1 for the one above the principal port). The measures in π are clearly oblivious of the changes. Let $\varrho = B[\sum_i \delta[\mu_i] + \pi]$, $\sigma = B[\delta[\lambda] + \pi]$, and



We have

$$\begin{aligned} !^e(p) &= (1 + \sum_j !(q_j)) (\sum_i \text{sp}^e(\delta[\mu_i]) + \sum_{\nu \in \pi} \text{sp}(\nu)) \\ &= (1 + \sum_j !(q_j)) (\text{sp}^e(\delta) \sum_i \text{sp}^e(\mu_i) + \sum_{\nu \in \pi} \text{sp}(\nu)) \\ &\leq (1 + \sum_j !(q_j)) (\text{sp}^\sigma(\delta) \sum_i \text{sp}^\sigma(\lambda) + \sum_{\nu \in \pi} \text{sp}(\nu)) = !^\sigma(p), \\ ?^e(q_h) &= ?(p)(1 + \sum_j !(q_j)) (\sum_i \text{sp}^e(\delta[\mu_i]) ?^e(r_h) + \sum_{\nu \in \pi} \text{sp}(\nu)) \\ &\leq ?(p)(1 + \sum_j !(q_j)) (\text{sp}^\sigma(\delta) (\sum_i \text{sp}^e(\mu_i)) ?^\sigma(r_h) + \sum_{\nu \in \pi} \text{sp}(\nu)) \\ &\leq (1 + \sum_j !(q_j)) (\text{sp}^\sigma(\delta) \text{sp}^\sigma(\lambda) ?^\sigma(r_h) + \sum_{\nu \in \pi} \text{sp}(\nu)) = ?^\sigma(q_h), \end{aligned}$$

so that $\varrho \preceq \sigma$. We can see here how the sum weighted with the spread value makes sure that the measures do not grow if the number of addends inside the box grows. As for the rest of the hypotheses, $\text{sp}(\varrho) = 1 = \text{sp}(\sigma)$, there are no new dormant wires and in fact

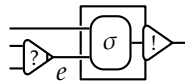
$$|\varrho| = \#(B) \|\sum_i \delta[\mu_i] + \pi\| < \#(B) \|\delta[\lambda] + \pi\| = |\sigma|,$$

using the base step. The box count depends solely on the variables outside the box. We can now apply inductive hypothesis, and get that

$$\|\omega[\sum_i \mu_i]\| = \|\psi[\varrho]\| < \|\psi[\sigma]\| = \|\omega[\lambda]\|.$$

Again replacing $<$ with \leq poses no problems. □

A short note about lax typing. We have used lax typing to define the measures. In fact the only point where it is relevant is when there is a clash between a $?$ active port and an auxiliary port of a box, that could be repaired by opening the box, like in the following example.



The catch is that we do not want to make the measure of the box contents dependent on the $?$ -weight of e , because it could possibly break or complicate the induction on which the definitions are based. Also there can be no interaction before the box is opened, and all steps on the box that are not dereliction ones bar the possibility of interaction for ever (as a contraction-contraction clash is created). Moreover the $!$ -weight on e^σ should appear in the $!$ one of e (as the box could be opened), but applying \mathbf{s} -conversion if σ is a sum would put a

second contraction on e , so $!(e^\sigma)$ would not appear anymore. Invariance under \sim -conversion would be broken, unless we overtly complicate the measures.

We indeed feel that the finite development theorem is valid also without lax typing: after all the configurations lax typing prohibits cannot be duplicated and interact with each other (as box linear or regular duplication renders the clash permanent). However, also because of the counterexample to the striction lemma shown in Section 4.3.4, which is our main reason for introducing lax typing, we feel it is right to restrict ourselves to reduction of laxly typed nets.

6.2.2 The Proof, Case by Case

Lemma 6.7. $\|\cdot\|$ has the following properties.

- if $\pi \xrightarrow{e} \pi'$ then $\|\pi'\| < \|\pi\|$;
- if $\pi \xrightarrow{m} \pi'$ then $\|\pi'\| \leq \|\pi\|$;
- if $\pi \sim \pi'$ then $\|\pi'\| = \|\pi\|$.

Because of the modularity lemma the proof of all the points above will boil down to checking the hypotheses on all pairs generating by context closure the relations above. Before getting to the cases, let us give the complete proof of the finiteness of developments supposing the above result.

Proposition 6.8 (SN of DiLL^o). *Reduction on DiLL^o is SN.*

Proof. Let us consider the measure given by $(\|\pi\|, \#_m(\pi) + \#_c(\pi))$ where $\#_m$ just counts the multiplicative cells in $c_!(\pi)$, and $\#_c$ weights coweakenings, weakenings and boxes containing 0 in the following way:

$$\#_c = \# \{ c \in c_!(\pi) \mid \sigma(c) = !_0 \} + \sum_{\substack{c \in c_!(\pi) \\ \sigma(c) = ?_0}} (1 + d(c)) + \sum_{\substack{B \in b_!(\pi) \\ \sigma(B) = 0}} 2 \deg(B)(1 + d(B)).$$

Then

- if $\pi \xrightarrow{e} \pi'$ the first component strictly decreases;
- if $\pi \xrightarrow{m} \pi'$ then $\|\pi'\| = \|\pi\|$, $\#_m(\pi') < \#_m(\pi)$ and $\#_c(\pi) = \#_c(\pi')$;
- if $\pi \xrightarrow{n} \pi'$ then a (co)weakening disappears, so $\#_c(\pi') < \#_c(\pi)$, and the rest is unchanged;
- if $\pi \xrightarrow{p} \pi'$ then either one or more weakenings have their depth strictly lowered, or, if we are creating a weakening out of a box B with $\sigma(B) = 0$, we have

$$1 + d(B) + 2(\deg(B) - 1)(1 + d(B)) = (1 + 2(\deg(B) - 1))(1 + d(B)) < 2 \deg(B)(1 + d(B)),$$

and the rest is unchanged;

- if $\pi \xrightarrow{z} \pi'$ then let B be the reduced box: 1 coweakening and $\deg(B) - 1$ weakenings at depth $d(B)$ are created, but

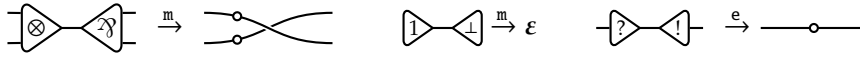
$$1 + (\deg(B) - 1)(1 + d(B)) \leq \deg(B)(1 + d(B)) < 2 \deg(B)(1 + d(B)),$$

while the rest remains unchanged;

- if $\pi \sim \pi'$ then all multiplicative, weakening and coweakening cells are in both polynets and at the same depth, so also $\#_c$ is constant. \square

We now begin dealing with the cases.

Multiplicative and dereliction against codereliction.



There is not much to say: all exponential measures clearly remain the same. The \circ -mark avoids that new exponential path be opened.

Dereliction against box.

$$\lambda := p \bullet \text{?} \begin{array}{c} d \\ \text{!} \end{array} \boxed{\sum_j \nu_j} \text{!} \text{?} \vec{q} \xrightarrow{e} \sum_j p \bullet \text{?} \begin{array}{c} \nu_j \end{array} \text{!} \text{?} \vec{q} =: \sum_i \mu_i$$

If $\sum_i \mu_i = 0$ then we are done, as $|\lambda| = [|d|] > []$. Suppose then that the contents are not 0. There are no new dormant wires.

Replacement:

$$\begin{aligned} !^{\mu_i}(p) &= 1 = !^\lambda(p), \\ ?^{\mu_i}(q_j) &= ?(q_j^{\nu_i}) \leq ?(p)(1 + \sum_j !(q_j)) \sum_i \text{sp}(\nu_i) ?(q^{\nu_i}) = ?^\lambda(q_j). \end{aligned}$$

Spread:

$$\sum_i \text{sp}(\mu_i) = \sum_i \text{sp}(\nu_i) \leq (1 + \sum_j !(q_j)) \sum_i \text{sp}(\nu_i) = !^\lambda(d) = \text{sp}(\lambda).$$

Weight:

$$|\mu_i| = |\nu_i| < [|d|] + (1 + \sum_j !(q_j)) \sum_i \text{sp}(\nu_i) |\nu_i| = |\lambda|.$$

Weakening against box.

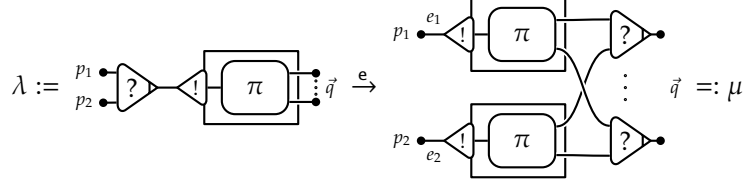
$$\lambda := \text{?} \begin{array}{c} d \\ \text{!} \end{array} \boxed{\sum_j \nu_j} \text{!} \text{?} \vec{q} \xrightarrow{e} \begin{array}{c} \text{?} \\ \vdots \\ \text{?} \end{array} \text{!} \text{?} \vec{q} =: \mu$$

No new dormant wires.

Replacement: $?^\mu(q_j) = 1 \leq ?^\lambda(q_j),$

Spread: $\text{sp}(\mu) = 1 = \text{sp}(\lambda),$

Weight: $|\mu| = [] < |\lambda|.$

Contraction against box.

The new dormant wires are e_1 and e_2 . Both the nets π in μ have exactly the same measures as π in λ (because of !-weight on auxiliary contraction ports is the same as that on the principal one).

Replacement:

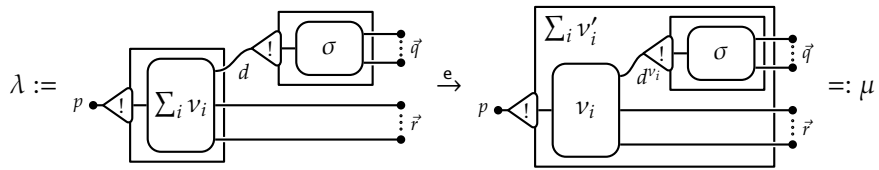
$$\begin{aligned} ?^\mu(q_j) &= ?(p_1)(1 + \sum_j !(q_j)) \sum_{\nu \in \pi} \text{sp}(\nu) ?(q^\lambda) \\ &\quad + ?(p_2)(1 + \sum_j !(q_j)) \sum_{\nu \in \pi} \text{sp}(\nu) ?(q^\lambda) \\ &= (?(p_1) + ?(p_2))(1 + \sum_j !(q_j)) \sum_{\nu \in \pi} \text{sp}(\nu) ?(q^\lambda) = ?^\lambda(q_j), \\ !^\mu(p_i) &= !^\lambda(d). \end{aligned}$$

Spread:

$$\text{sp}(\mu) = 1 = \text{sp}(\lambda).$$

Weight:

$$\begin{aligned} |e_i|^\mu &= ?^\mu(p_i) + !^\mu(p_i) < (?^\lambda(p_1) + ?^\lambda(p_2)) !^\lambda(d) = |d|, \\ |\mu| + [|e_1|, |e_2|] &= [|e_1|, |e_2|] + ?(p_1)(1 + \sum_j !(q_j)) \|\pi\|^\mu + ?(p_2)(1 + \sum_j !(q_j)) \|\pi\|^\mu \\ &< [|d|] + (?(p_1) + ?(p_2))(1 + \sum_j !(q_j)) \|\pi\|^\lambda = |\lambda|. \end{aligned}$$

Box against box.

No new dormant wires. All measures inside both boxes are constant during this reduction. We suppose $\sum_i \nu_i \neq 0$, or else the step is trivial. Let ν'_i be the addend inside the second box in μ , corresponding to ν_i . If B is the box with the ν_i s inside in both nets then

$$\begin{aligned} \#^\lambda(B) &= ?(p)(1 + !^\lambda(d) + \sum_k (r_k)) = ?(p)(1 + (1 + \sum_h !(f_h)) \text{sp}(\sigma) + \sum_k !(r_k)) \\ &> ?(p)(1 + \sum_h !(f_h) + \sum_k !(r_k)) = \#^\mu(B), \end{aligned}$$

where $\text{sp}(\sigma) = 1$ if $\sigma = 0$, and the sum of its spreads otherwise.

Replacement:

$$\begin{aligned}
?^\mu(q_j) &= \#^\mu(B) \sum_i \text{sp}(\nu'_i) ?^\mu(q^{\nu'_i}) \\
&= \#^\mu(B) \sum_i (\text{sp}(\nu_i) ?^\mu(d^{\nu_i}) (1 + \sum_h !(q_h)) \sum_{\kappa \in \sigma} ?^\mu(q^\kappa)) \\
&< (\#^\lambda(B) \sum_i \text{sp}(\nu_i) ?^\lambda(d^{\nu_i})) (1 + \sum_h !(q_h)) \sum_{\kappa \in \sigma} ?^\lambda(q^\kappa) \\
&= ?^\lambda(d) (1 + \sum_h !(q_h)) \sum_{\kappa \in \sigma} ?^\lambda(q^\kappa) = ?^\lambda(q_j), \\
?^\mu(r_j) &= \#^\mu(B) \sum_i \text{sp}(\nu'_i) ?^\mu(r^{\nu'_i}) < \#^\lambda(B) \sum_i \text{sp}(\nu_i) ?^\lambda(r^{\nu_i}) = ?^\lambda(r_j).
\end{aligned}$$

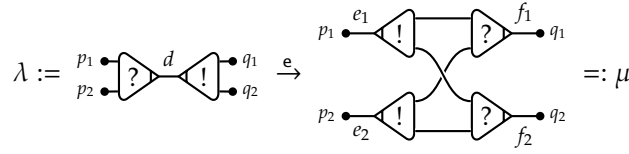
Spread:

$$\text{sp}(\mu) = 1 = \text{sp}(\lambda).$$

Weight:

$$\begin{aligned}
|d^{\nu_i}|^\mu &= ?^\mu(d^{\nu_i}) + !^\mu(d^{\nu_i}) < \#^\lambda(B) \sum_i \text{sp}(\mu_i) ?^\lambda(d^{\nu_i}) + !^\lambda(d) = |d|^\lambda, \\
|\mu| &\leq \#^\mu(B) \sum_i \text{sp}(\nu'_i) (|\nu_i| + [|d^{\nu_i}|^\mu] + ?^\mu(d^{\nu_i}) (1 + \sum_h !(q_h)) \|\sigma\|) \\
&< \#^\lambda(B) \sum_i \text{sp}(\nu_i) [|\nu_i|] + \#^\lambda(B) \sum_i \text{sp}(\nu_i) |\nu_i| \\
&\quad + (\#^\lambda(B) \sum_i \text{sp}(\nu_i) ?^\lambda(d^{\nu_i})) (1 + \sum_h !(q_h)) \|\sigma\| \\
&< |d|^\lambda + \#^\lambda(B) \|\sum_i \nu_i\| + ?^\lambda(d) (1 + \sum_h !(q_h)) \|\sigma\| = |\lambda|.
\end{aligned}$$

Contraction against cocontraction



The new dormant wires are all e_1 , e_2 , f_1 and f_2 .

Replacement:

$$!^\mu(p_i) = !(q_1) + !(q_2) = !^\lambda(p_i), \quad ?^\mu(q_i) = ?(p_1) + ?(p_2) = ?^\lambda(q_i).$$

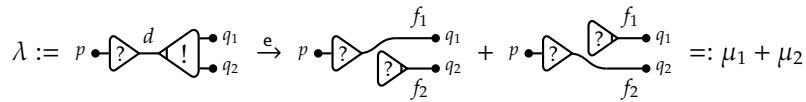
Spread:

$$\text{sp}(\mu) = 1 = \text{sp}(\lambda).$$

Weight:

$$\begin{aligned}
|e_i|^\mu &= !^\mu(p_i) (?^\mu(q_1) + ?^\mu(q_2)) < (!^\lambda(p_1) + !^\lambda(p_2)) (?^\lambda(q_1) + ?^\lambda(q_2)) = |d|^\lambda, \\
|f_i|^\mu &= ?^\mu(q_i) (!^\mu(p_1) + !^\mu(p_2)) < (?^\lambda(q_1) + ?^\lambda(q_2)) (!^\lambda(p_1) + !^\lambda(p_2)) = |d|^\lambda, \\
|\mu| + [|e_1|, |e_2|, |f_1|, |f_2|] &< [] + |d|^\lambda = |\lambda|.
\end{aligned}$$

Dereliction against cocontraction. (and codereliction against contraction)



f_1 and f_2 are the new dormant wires.

Replacement:

$$!^{\mu_i}(p) = 1 = !^\lambda(p), \quad ?^{\mu_i}(q_j) = 1 = ?^\lambda(q_j).$$

Spread:

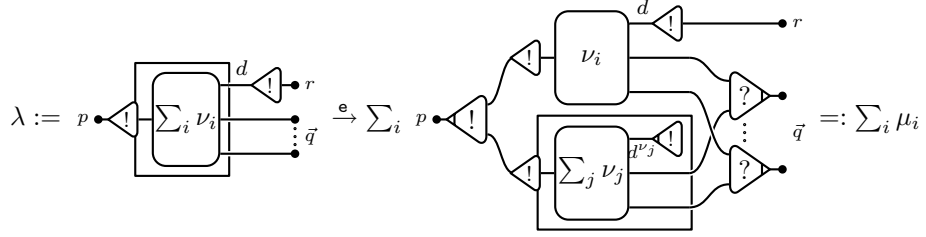
$$\text{sp}(\mu_1) + \text{sp}(\mu_2) = !(q_1) + !(q_2) = \text{sp}(\lambda).$$

Weight:

$$|\mu| + [|f_1|, |f_2|] = [1 + !(q_1), 1 + !(q_2)] < [1 + !(q_1) + !(q_2)].$$

Dereliction against coweakening. Trivial, as it is a reduction to 0, and the same for codereliction against weakening.

Codereliction against box.



No new dormant wires. We may suppose $\sum_i \nu_i \neq 0$, as otherwise it is trivial. In μ_i both the ν_i outside the box and the ν_j s inside get the same measures. If B is the box, then

$$\#^\lambda(B) = ?(p)(2 + \sum_h !(q_h)) = ?(p) + \#^\mu(B) \geq 1 + \#^\mu(B).$$

Replacement:

$$\begin{aligned} !^{\mu_i}(p) &= 1 + (1 + \sum_h !(q_h)) \sum_j \text{sp}(\nu_j) \leq (2 + \sum_h !(q_h)) \sum_j \text{sp}(\nu_j) = !^\lambda(p), \\ ?^{\mu_i}(q_h) &= ?^{\mu_i}(q_h^{\nu_i}) + \#^{\mu_i}(B) \sum_j \text{sp}(\nu_j) ?^\mu(q_h^{\nu_j}) \\ &\leq \sum_j \text{sp}(\nu_j) ?^\lambda(q_h^{\nu_j}) + \#^{\mu_i}(B) \sum_j \text{sp}(\nu_j) ?^\lambda(q_h^{\nu_j}) \\ &\leq \#^\lambda(B) \sum_j \text{sp}(\nu_j) ?^\lambda(q_h^{\nu_j}) = ?^\lambda(q_h), \\ ?^\mu(r) &= 1 = ?^\lambda(r). \end{aligned}$$

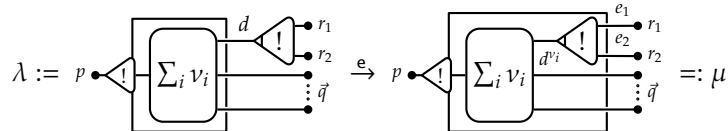
Spread:

$$\begin{aligned} \sum_i \text{sp}(\mu_i) &= \sum_i ?(p) \text{sp}(\nu_i) ?^{\mu_i}(d) = ?(p) \sum_i \text{sp}(\nu_i) ?^\lambda(d^{\nu_i}) \\ &\leq ?(p)(1 + \sum_h !(q_h)) \sum_i \text{sp}(\nu_i) ?^\lambda(d^{\nu_i}) = ?^\lambda(d) = \text{sp}(\lambda). \end{aligned}$$

Weight:

$$\begin{aligned} |d|^{\mu_i} &= ?^{\mu_i}(d) + 1 = ?^\lambda(d^{\nu_i}) + 1 < ?^\lambda(d) + 1 = |d|, \\ |d^{\nu_j}|^{\mu_i} &= ?^{\mu_i}(d^{\nu_j}) + 1 = ?^\lambda(d^{\nu_j}) + 1 < |d|, \\ |\mu_i| &\leq |\nu_i| + [|d|^{\mu_i}] + \#^\mu(B) \sum_j \text{sp}(\nu_j) (|d^{\nu_j}| + |\nu_j|) \\ &\leq [|d|^{\mu_i}] + \#^\mu(B) \sum_j \text{sp}(\nu_j) [|d^{\nu_j}|] + (1 + \#^\mu(B)) \sum_j \text{sp}(\nu_j) |\nu_j| \\ &< [|d|^\lambda] + \#^\lambda(B) \left\| \sum_j \nu_j \right\| = |\lambda|. \end{aligned}$$

Cocontraction against box



As usual, we consider only the case in which $\sum_i \nu_i \neq 0$. Let B be the box, and e_1 and e_2 the new dormant wires.

Replacement:

$$\begin{aligned} !^\mu(p) &= (1 + \sum_h !(f_h)) \sum_i \text{sp}(\nu_i) = !^\lambda(p), \\ ?^\mu(q_h) &= \#^\mu(B) \sum_i \text{sp}(\nu_i) ?^\mu(q^{\nu_i}) = ?^\lambda(q_h), \\ ?^\mu(r_j) &= \#^\mu(B) \sum_i \text{sp}(\nu_i) ?^\mu(d^{\nu_i}) = ?^\lambda(r_j). \end{aligned}$$

Spread:

$$\text{sp}(\mu) = 1 = \text{sp}(\lambda).$$

Weight:

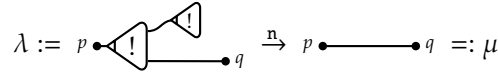
$$\begin{aligned} |d^{\nu_j}|^\mu &= ?^\mu(d^{\nu_j}) + !(r_1) + !(r_2) \leq \#^\lambda(B) \sum_j \text{sp}(\nu_j) ?^\lambda(d^{\nu_j}) + !(r_1) + !(r_2) = |d|, \\ |\mu| &\leq \#^\mu(B) \sum_j \text{sp}(\nu_j) (|d^{\nu_j}|^\mu + |\nu_j|) = \#^\lambda \sum_j \text{sp}(\mu_j) [|d^{\nu_j}|^\mu] + \#^\lambda(B) \left\| \sum_j \nu_j \right\| \\ &< [|d|] + \#^\lambda(B) \left\| \sum_j \nu_j \right\| = |\lambda|. \end{aligned}$$

Coweakening against box. No different, if easier, than the previous step.

Now we check that the measure is invariant under equivalence and non increasing for the corresponding reductions. The modularity lemma can be applied a second time exchanging the roles of λ and μ , so it can be used also for checking equality.

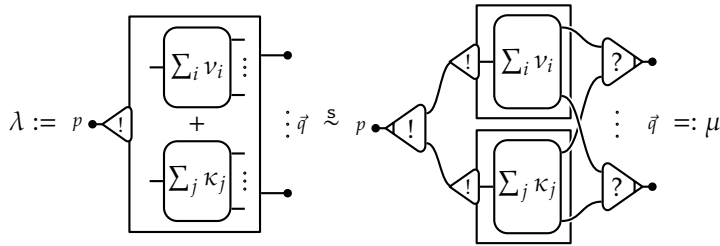
Associative conversion. This is completely straightforward, as the operation on weights assigned to (co)contractions is sum, which is associative.

Neutral reduction.



Clearly $!^\mu(p) < 1 + !(q) = !^\lambda(p)$ and $?^\mu(q) = ?^\lambda(q)$, which suffices.

Bang sum conversion.



The conversion does not create new dormant wires. Moreover if B is the box on the left, then $\#^\lambda(B)$ is the count also of both boxes on the right. Also the contents of the boxes, which we denote by B' and B'' , get the same measures. Recall that such contents are required to be non zero.

Replacement:

$$\begin{aligned} ?^\mu(q_h) &= \#^\mu(B) (\sum_i \text{sp}(\nu_i) ?^\mu(q^{\nu_i}) + \sum_j \text{sp}(\kappa_j) ?^\mu(q^{\kappa_j})) = ?^\lambda(q_h), \\ !^\mu(p) &= (1 + \sum_h !(q_h)) (\sum_i \text{sp}(\nu_i) + \sum_j \text{sp}(\kappa_j)) = !^\lambda(p), \end{aligned}$$

Spread:

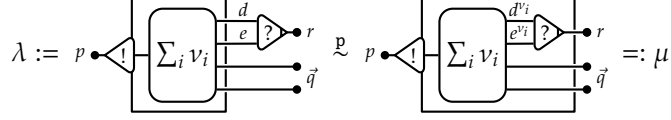
$$\text{sp}(\mu) = 1 = \text{sp}(\lambda);$$

Weight:

$$|\mu| = \#^\mu(B') \|\sum_i \nu_i\| + \#^\mu(B'') \|\sum_j \kappa_j\| = \#^\lambda(B) (\|\sum_i \nu_i\| + \|\sum_j \kappa_j\|) = |\lambda|$$

Bang zero reduction. Straightforward check.

Push equivalence.



Here we have only to check for $?(r)$, as the rest is trivial.

$$\begin{aligned} ?^\mu(r) &= \#^\mu(B) \sum_i \text{sp}(\nu_i) (?^\mu(d^{\nu_i}) + ?^\mu(e^{\nu_i})) \\ &= \#^\lambda(B) \sum_i \text{sp}(\nu_i) ?^\lambda(d^{\nu_i}) + \#^\lambda(B) \sum_i \text{sp}(\nu_i) ?^\lambda(e^{\nu_i}) = ?^\lambda(r). \end{aligned}$$

Pull reduction. The total extraction, which does not split the box, is a straightforward simplification of the above point. The partial one can be recovered from the total applying an **s**-conversion, a total **p**-reduction and a number of **n**-reductions. Combining these steps we can infer the decrease of the measures. \square

6.3 Confluence of Developments

Recall that \sim and **c** may be **a**-equivalence and **n**-reduction, or full **asp**-equivalence and **nzp**-reduction. We will see we cannot separate **s**-equivalence from the **p** one (and similarly for their associated reductions).

Proposition 6.9. *Reduction on DiLL° is $\text{CR}\sim$. Moreover:*

- **ec** is $\text{CR}\sim$;
- **m** is strongly confluent;
- **ec** commutes with **m**, i.e. $\xrightarrow{\mathbf{m}*\mathbf{ec}^*} \subseteq \xrightarrow{\mathbf{ec}^*}\xrightarrow{\mathbf{m}^*}$.

The last point can be furtherly strengthened by $\xrightarrow{\mathbf{m}*\mathbf{ec}} \subseteq \xrightarrow{\mathbf{ec}}\xrightarrow{\mathbf{m}^*}$.

Notice that, as expected, the two points about multiplicative reduction do not rely on \sim . The proof relies on the strong normalization property of DiLL° we proved in the previous section, and the analog of Newman's lemma in reduction modulo, Lemma 4.2. We therefore have to prove the two remaining hypotheses, as usual by going through the cases: local confluence modulo \sim (Lemma 6.13), and local coherence with the generating relation \vdash . Before going into that we first cut on the critical pairs to look at, and then settle the two easy points of the proposition.

Lemma 6.10. *If $\pi' \xrightarrow{x} \pi \xrightarrow{\mathbf{ec}} \pi''$ with the left reduction being one of any kind on a redex contained in a box B , and the right one a reduction on B , then π' and π'' are joinable by $\pi' \xrightarrow{\mathbf{ec}} \xrightarrow{x^*} \pi''$, with the sole exception of the left reduction being an **e** one turning an addend of $\sigma(B)$ to 0, and right one being a pull reduction on that addend. In this very special case we can join by $\pi' \xrightarrow{\mathbf{e}} \xrightarrow{\mathbf{c}^*} \xrightarrow{\mathbf{p}^=} \pi''$.*

Proof. Let $\xrightarrow{1}$ and $\xrightarrow{2}$ be the two diverging reductions. The special case is the only one in which the redex of $\xrightarrow{2}$ is deleted by $\xrightarrow{1}$. In every other cases firing $\xrightarrow{2}$ the contents of B are either deleted or glued in one or two copies in the reduct (after which the copies may be even more due to additive “duplication”), and the same happens when we fire the unique redex of $\xrightarrow{2}$ in π' . We may then fire all copies of the redex of $\xrightarrow{1}$ in π'' and join the peak, without even relying on \sim .

As for the special case, if the \mathbf{p} reduction is total the confluence diagram is trivial, as we can still extract a weakening from B if $\sigma(B)$ turns to 0. For the partial one, if the detached box in the \mathbf{p} -reduct gets its contents reduced to 0, then we can make such box disappear by a \mathbf{z} -reduction and several \mathbf{n} ones, reducing to π'' . \square

We get as a direct consequence the commutation of \mathbf{ec} and \mathbf{m} , for the whole of DiLL in effect.

Lemma 6.11. *We have $\xrightarrow{\mathbf{m} \mathbf{ec}} \subseteq \xrightarrow{\mathbf{ec} \mathbf{m}^*}$, which in particular implies $\xrightarrow{\mathbf{m}^* \mathbf{ec}^*} \subseteq \xrightarrow{\mathbf{ec}^* \mathbf{m}^*}$.*

Proof. For the first part, suppose $\pi' \xrightarrow{\mathbf{m}} \pi \xrightarrow{\mathbf{ec}} \pi''$. If the redexes are completely orthogonal then $\pi' \xrightarrow{\mathbf{ec}} \xrightarrow{\Sigma \mathbf{m}} \pi''$: the \mathbf{ec} reduction may introduce sums (see page 37 for the definition of the sum reduction), while the \mathbf{m} cannot. If they are not orthogonal then they must fall in the hypotheses of the above lemma and we again get $\xrightarrow{\mathbf{ec} \mathbf{m}^*}$.

For the implication, it is a direct consequence of a lemma by Huet [Hue80], stating that given two reductions $\xrightarrow{1}$ and $\xrightarrow{2}$ such that $\xrightarrow{1} \xrightarrow{2} \subseteq \xrightarrow{2} \xrightarrow{1^*}$ then the two reductions commute. \square

As usual, dealing with $\xrightarrow{\mathbf{m}}$ is straightforward.

Lemma 6.12. *The \mathbf{m} reduction is strongly confluent.*

Proof. Direct consequence of Proposition 2.12 for interaction nets. \square

6.3.1 Local Confluence Modulo

So finally we embark on the proof of local confluence of \mathbf{ec} alone.

Lemma 6.13. *The \mathbf{ec} reduction is locally confluent modulo \sim in DiLL $^\circ$.*

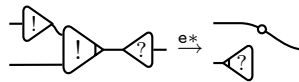
Here \mathbf{c} and \sim can either be \mathbf{n} and \mathfrak{A} alone, or the full ones.

Because of Lemma 6.10, we need only to look for peaks happening at the same depth. Most of the numerous critical pairs are straightforward and left to the reader. We just give a complete list with a short description for those².

- Box on box on dereliction, weakening or contraction, or two boxes on a third: these are in LL and therefore are known. The one with dereliction just needs to take into account the sums that may have arisen. One just takes uses the $\xrightarrow{\Sigma \mathfrak{e}}$ version of the steps taken in LL.
- Any combination of substituting boxes with coweakenings and cocontractions in the cases above: coweakenings and cocontractions on a box behave the same way as a box, so the confluence diagrams are identical.

²More cases will be added in an imminent revision of the thesis.

- Coderelection and either box, coweakening or cocontraction on box: easy by duplicating the box, coweakening or cocontraction with the contraction created by the coderelection, and making a copy enter inside the box (all this on each addend created by the coderelection).
- Coderelection on box on weakening: both sides of the peak reduce to 0.
- Neutral reduction against a reduction on the relative (co)contraction: joinable using (co)weakening steps.
- z reducible box against (co)dereliction: both sides reduce to 0, either by opening (a linear copy of) the 0 box or by (co)dereliction against (co)weakening.
- z reducible box on contraction or weakening: easy, the former needs n -reductions.
- z reducible box on box: straightforward, but needs a total pull reduction, as if we z -reduce the box inside we need to get the created weakenings outside.
- Every other reduction on a z -reducible box: easy, anything entering a 0 box disappears, but so does when cutting against a weakening.
- Total pull reduction and a reduction on the same box but not the same wire: straightforward by pulling all the residuals of the weakening (or leaving them be if the box got opened).
- Total pull reduction and a coderelection on the same wire: both reduce to 0.
- Total pull reduction and any other cell on the same wire: straightforward, the result of reducing a weakening inside or outside is the same, provided we then pull all the produced weakenings out.
- Partial pull reduction: using the same reasoning we will do for s -equivalence in the proof of Lemma 6.14, together with the above one gets all the confluence diagrams needed.
- Coderelection on box on dereliction: the confluence diagram is shown in Figure 6.2. We use the fact that



as the other addend reduces to 0. Notice that we must use the neutral reduction of weakening.

- Coderelection on box on contraction: Figure 6.3 shows this confluence diagram. Here we need both the neutral rule of coweakening and associativity of contraction.
- Two coderelections on box: the resulting reduction is shown in Figure 6.4. The two diverging one step reductions are joined symmetrically, but only if we have associativity of (co)contractions. \square

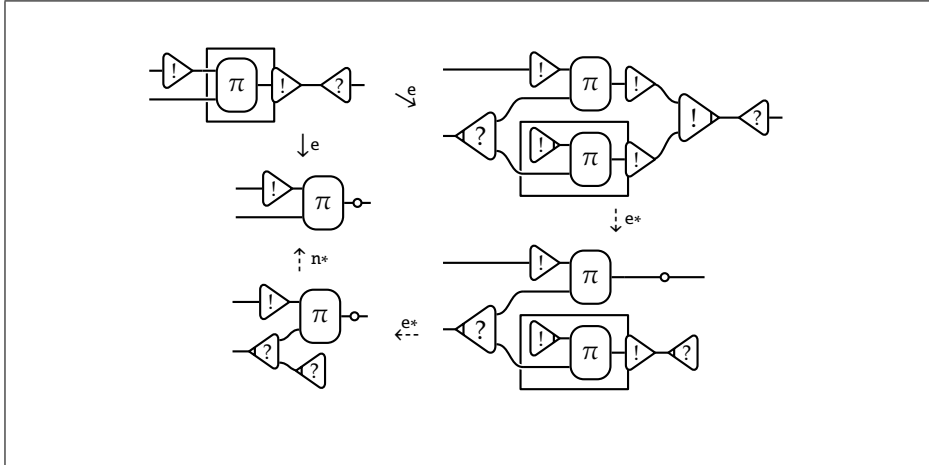


Figure 6.2: Confluence diagram of codereliction on box on dereliction.

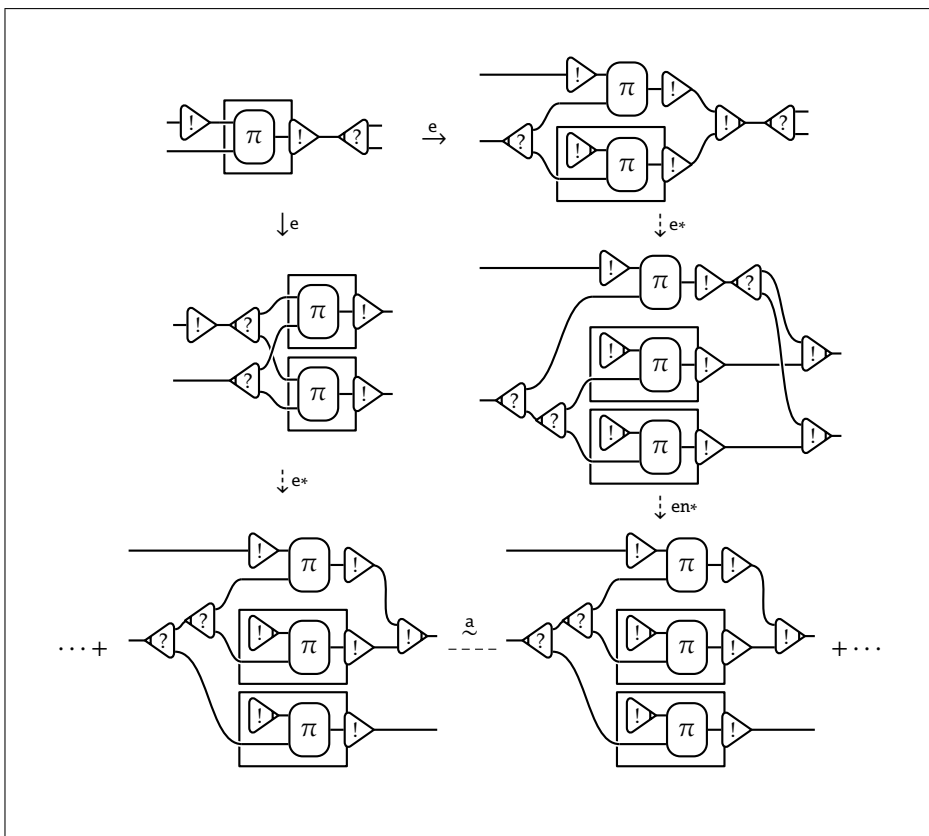


Figure 6.3: Confluence diagram of codereliction on box on contraction. The $+ \dots$ part indicates a symmetric addend.

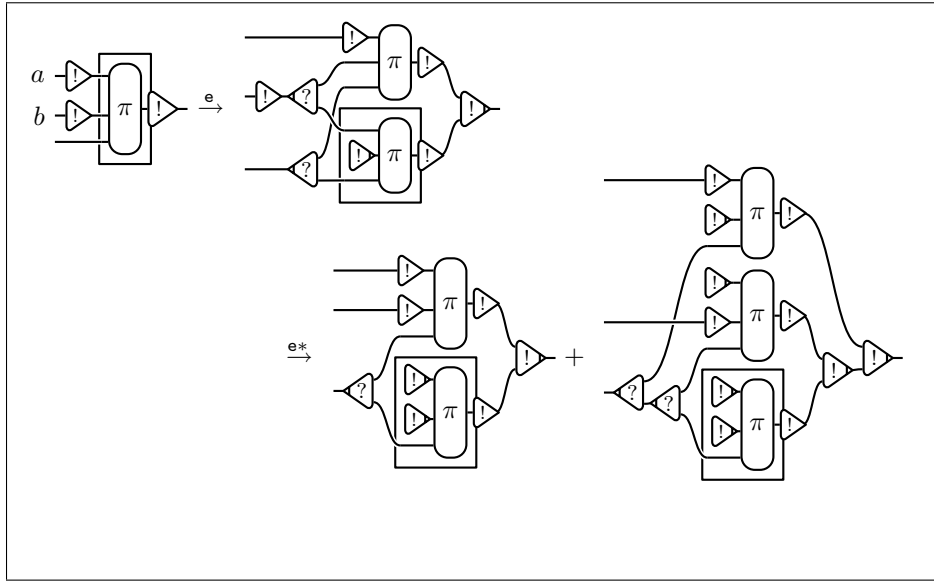


Figure 6.4: Reduction of a box with two coderelictions on it. Starting with codereliction b swaps the two linear copies of the box contents and therefore both the cocontraction and contraction trees in the last addend.

6.3.2 Local Coherence

Lemma 6.14. *The \mathbf{ec} reduction is locally coherent with both $\stackrel{\mathbf{a}}{\vdash}$ and $\stackrel{\mathbf{asp}}{\vdash}$ in \mathbf{DiLL}° .*

Lemma 6.10 applies also by replacing the \mathbf{ec} -reduction by a \vdash conversion, except the reduction to zero, given that all the equivalences are explicitly forbidden on boxes containing 0. Also the \mathbf{p} -conversion forms a critical pair with a reduction deleting a contraction by neutrality.

The actual confluence diagrams are left to the reader. Here is a complete list with sketched explanations.

- All critical pairs with associativity: trees of contractions and cocontractions are known to behave like generalized n -ary (co)contractions. In particular when dealing with a tree of two nested (co)contraction, it suffices to complete the reduction on both to join the peak. Neutrality on associativity is trivial.
- Box, coweakening or cocontraction on a \mathbf{s} conversion: on one side they enter the box getting additively duplicated inside, while on the other they are duplicated by a contraction before entering. In any case another \mathbf{s} conversion closes
- Dereliction on a \mathbf{s} conversion: on one side the net is additively split by opening the box, on the other the net is split first by the dereliction on cocontraction rule, then depending on the box chosen the other gets deleted, and finally closing with \mathbf{n} -reductions we obtain the same net.
- Contraction or weakening on a \mathbf{s} conversion: both traverse the two sides doing the same operations, on one side by being duplicated by the leading cocontraction.

- **s**-convertible box on another box: both sides may enter the box, safe for the trailing contractions, which must be settled by a push conversion.
- Codereliction on a **s** conversion: this diagram is shown in Figure 6.5.
- Reduction to zero taking an addend of the conversion: on the side where the box is split, we **z**-reduce the box and by **n**-reductions we get exactly to the other side.
- Pull reduction on **s**-conversion: the pull is designed to integrate a sort of **s**-conversion step, so that it may be always performed on two sides of a conversion. **n**-conversion does the rest.
- Box, coweakening or cocontraction on a **p** conversion: the cell gets duplicated and enters the box on both sides, but in opposite order.
- Dereliction on a **p** conversion: the reductions on the two sides are equal.
- Contraction or weakening on a **p** conversion: the way in which contractions are stacked may change, but **a**-conversion (and **n**-reduction) ensures joinability.
- **p**-convertible box on another box: apart from having to exit/enter two boxes on one side, the two are the same.
- Codereliction on a **p** conversion: additive splitting and box opening happen on both sides, but in different order.
- Reduction to zero of the **p**-converted box: by **c**-normalization we get just weakenings and coweakenings on both sides.
- Pull reduction on **p**-conversion, on different auxiliary ports: we can **p**-convert all boxes involved, we just have to reorganize the stack of contractions by **a**-conversion.
- Pull reduction on **p**-conversion, with the weakening on an auxiliary port interested by the conversion: pull and neutral reductions join the two sides.
- Neutral reduction on **p**-conversion: **n**-reduction can block the conversion on one side by destroying the contraction of an addend. This is joined by a possibly partial **p**-reduction and then by a full **asp** conversion. \square

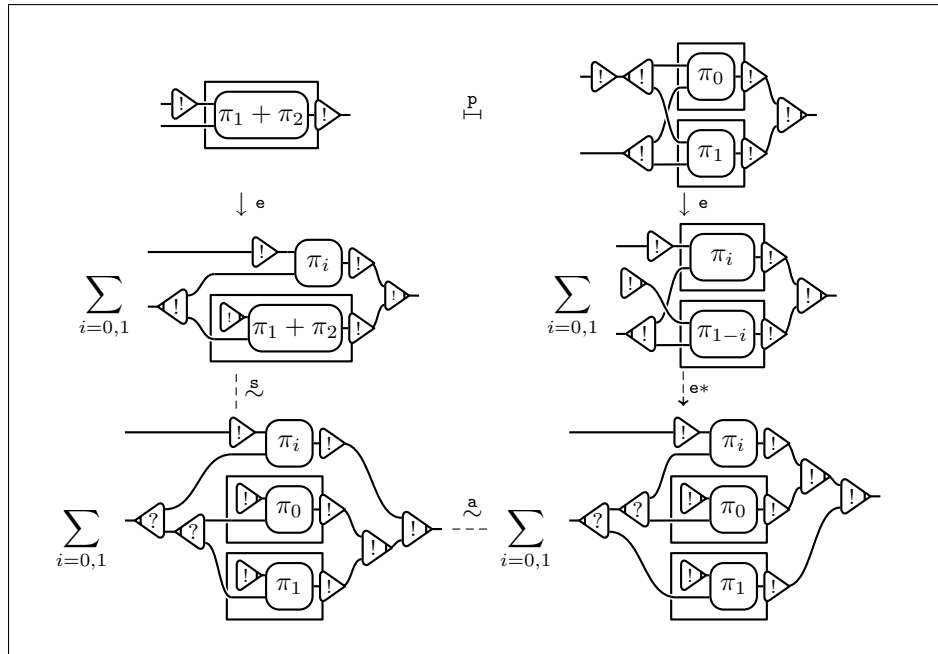


Figure 6.5: Coherence diagram between the bang sum equivalence and a codereliction.

Chapter 7

Standardization and Conservation Theorems

In this chapter we will prove other fundamental properties borrowed from λ -calculus and LL. Following Barendregt's terminology for λ -calculus, as found in [Bar84], we will prove standardization (Theorem 7.1) and conservation (Theorem 7.2). The first states that each reduction chain can be turned into a *standard* which, in LL terms, proceeds in order of depth. The latter states that *non erasing* reduction (in λ -calculus those redexes where the bound variable occurs at least once) conserves infinite reductions, and that weak normalization for non erasing reductions is therefore equivalent to strong normalization of the usual one. Like the finite developments for confluence, the conservation theorem is a stepping stone for results of strong normalization, with the advantage of being set in the (essentially) untyped case (see [PTdF08]).

First of all let us define what exactly **erasing** reductions are. These must clearly contain the following one:

- weakening on box (this was the only erasing one in MELL);
- all those reducing to 0, namely weakening (resp. coweakening) on coderelection (resp. dereliction) and (co)dereliction on a box containing 0;
- all reductions which erase a box, cocontraction or coweakening by making them enter a box containing 0;

For a number of technical reasons, we generalize and add in more reductions. We denote by \xrightarrow{er} all the ones reducing redexes containing at depth 0 a weakening, coweakening or box containing 0, together with all the canonical reductions. By $\xrightarrow{\neg er}$ (**non erasing** reduction) we denote all the other ones.

Standardization is broken by **ps**-conversions and **pz**-reductions (as is in LL for the push and pull rewriting rules), and we need it also for non erasing reductions in the proof of the conservation theorem. Therefore just in the following statement let \rightarrow denote either $\xrightarrow{\neg er}$ or \xrightarrow{mec} with **c** just the neutral reduction and \sim just the **a**-equivalence.

Theorem 7.1 (Standardization). *Given a laxly typed DiLL proof net π , if $\pi \xrightarrow{*} \pi'$, where then there is a reduction chain (called standard) from π to π' such that if c_i is the sequence of cuts fired, the depth $d(c_i)$ is monotone increasing.*

The following is valid for the full reduction.

Theorem 7.2 (Conservation). *Given a laxly typed DiLL proof net π , it is strongly normalizing modulo \sim if and only if it is weakly normalizable for non erasing steps.*

Combining the result by Pagani of weak normalization for simply typed DiLL contained in [Pag09] we get the following.

Corollary 7.3. *Simply typed DiLL proof nets are strongly normalizing modulo \sim .*

The next section will be devoted to the proof of the standardization theorem, which mirrors from a distance the corresponding one for λ -calculus as shown in [Bar84]. In particular it makes use of the finite developments theorem.

For the conservation property the basic idea is to adapt the proof of the same result for LL [PTdF08]. So we want to use the method employed by Gandy in his proof of normalization of Gödel's system T, based on an increasing measure. In fact, Gandy's work is based on a lemma usually attributed to Nederpeld.

Lemma 7.4 (Nederpeld). *Let (A, \rightarrow) be an abstract reduction system such that*

- \rightarrow is confluent;
- there is a measure $\|\cdot\|$ such that $t \rightarrow t'$ implies $\|t\| < \|t'\|$.

Then t is weakly normalizable if and only if it is strongly so.

The following is the same result adapted to reduction modulo.

Lemma 7.5. *Let (A, \rightarrow) be an abstract reduction system with an equivalence relation \sim on A such that*

- \rightarrow is Church Rosser modulo \sim ;
- there is a measure $\|\cdot\|$ such that $t \rightarrow t'$ implies $\|t\| < \|t'\|$ and $t \sim t'$ implies $\|t\| = \|t'\|$

Then t is weakly normalizable if and only if it is strongly normalizable modulo \sim .

Proof. Suppose t is WN, with a normal form n (unique modulo \sim). For every t' with $t \xrightarrow{*} t'$ we have $t' \xrightarrow{*} n$ by CR \sim , and thus $\|t'\| \leq \|n\|$. Therefore no reduction from t can infinitely increase $\|\cdot\|$. In fact no reduction can be longer than $\|\text{NF}(t)\| - \|t\|$. \square

Gandy's method amounts to applying this lemma to non erasing reduction, by enriching the language with increasing counters. The final result is achieved by applying a delaying lemma stating that erasing steps can always be postponed.

DiLL however suffers from a serious drawback, contrary to what happens in LL: the confluence property fails for non erasing reduction. In fact the same net shown in Figure 4.12 to justify lax typing, shows how confluence fails. The unjoinable critical peak is the one of dereliction on box on codereliction, as shown in Figure 6.2. One can notice there that the erasure of a box is definitely needed, in fact together with a reduction to 0 of an addend.

In Section 7.2 we will therefore present a modified version of the system, DiLL_{∂_0} , where the roles of dereliction and codereliction are replaced by new cells and where non erasing reduction is confluent. Section 7.3 uses Gandy's method on DiLL_{∂_0} , and a final lemma (Lemma 7.26) is used to then transfer conservation to DiLL .

7.1 Proof of the Standardization Theorem

For the remainder of this section only, \rightarrow denotes $\xrightarrow{\text{er}}$ or $\xrightarrow{\text{mec}}$, with just **n**-reduction and **a**-conversion. In the following let us denote by \rightarrow_0 (resp. $\rightarrow_{>0}$) a \rightarrow step happening at depth 0 (resp. greater than 0).

Lemma 7.6. *If in DiLL° we have $\pi \xrightarrow{*} \sigma$, then $\pi \xrightarrow{*}_0 \xrightarrow{*}_{>0} \sigma$.*

Proof. Using the finite developments theorem (Theorem 6.1), we can reason by well founded induction on π . If $\pi = \sigma$ we are done. Otherwise by inductive hypothesis we have $\pi \xrightarrow{\text{er}} \xrightarrow{*}_0 \xrightarrow{*}_{>0} \sigma$. We can suppose the reduction is $\pi \xrightarrow{+}_{>0} \xrightarrow{+}_0 \xrightarrow{*}_{>0} \sigma$, otherwise we are finished. If we thus show that $\xrightarrow{+}_{>0} \rightarrow_0 \subseteq \rightarrow_0 \xrightarrow{\text{er}*}$ we can conclude: we get $\pi \rightarrow_0 \pi'' \xrightarrow{\text{er}*} \sigma$, and applying again the inductive hypothesis on π'' gives the needed shape of reduction.

The only interesting case is when \rightarrow_0 touches a box B and $\xrightarrow{+}_{>0}$ happens inside it, as otherwise they are orthogonal and reorderable. In such cases it is straightforward by inspection of the rules that the 0-depth redex is present at the start and we have $\xrightarrow{+}_{>0} \rightarrow_0 \subseteq \rightarrow_0 \xrightarrow{\text{er}*}$ by firing all the copies of the redex inside B , where such copies may have an increased or decreased depth. **a**-conversions do not interfere with this reasoning. \square

Lemma 7.7. *Also in DiLL , if $\pi \xrightarrow{*} \sigma$, then $\pi \xrightarrow{*}_0 \xrightarrow{*}_{>0} \sigma$.*

Proof. Let \leftrightarrow (resp. \leftrightarrow_0 and $\leftrightarrow_{>0}$) be the parallel version of \rightarrow (resp. for depth 0 and greater than 0, see page 111). Namely, $\lambda \leftrightarrow \mu$ if and only if $\lambda \xrightarrow{*} \mu'$ in DiLL° and μ is μ' without the marks (and similarly for \leftrightarrow_0 and $\leftrightarrow_{>0}$). Now notice the following:

- $\leftrightarrow_{>0} \leftrightarrow_0 \subseteq \leftrightarrow$: no reduction inside \leftrightarrow_0 can be blocked by a mark created by $\leftrightarrow_{>0}$;
- Lemma 7.6 directly assures that $\leftrightarrow \subseteq \leftrightarrow_0 \leftrightarrow_{>0}$, so $\leftrightarrow_{>0} \leftrightarrow_0 \subseteq \leftrightarrow_0 \leftrightarrow_{>0}$.

It is then immediate, by iterating such inclusion, that

$$(\leftrightarrow_0 \cup \leftrightarrow_{>0})^* \subseteq (\leftrightarrow)^* (\leftrightarrow_{>0})^* = \xrightarrow{*}_0 \xrightarrow{*}_{>0}.$$

As

$$\xrightarrow{\text{er}} = \rightarrow_0 \cup \rightarrow_{>0} \subseteq \leftrightarrow_0 \cup \leftrightarrow_{>0},$$

we finally get that $\xrightarrow{\text{er}*} \subseteq \xrightarrow{*}_0 \xrightarrow{*}_{>0}$. \square

Proof of Theorem 7.1. We prove that $\pi \xrightarrow{*} \sigma$ can be turned into a standard reduction by induction on the depth of σ . First, apply Lemma 7.7 to get $\pi \xrightarrow{*}_0 \xrightarrow{*}_{>0} \sigma$. We notice here that π' has the same structure at depth 0 than σ , i.e. they are equal up to the contents of boxes at depth 0. Let μ_i be all the polynets

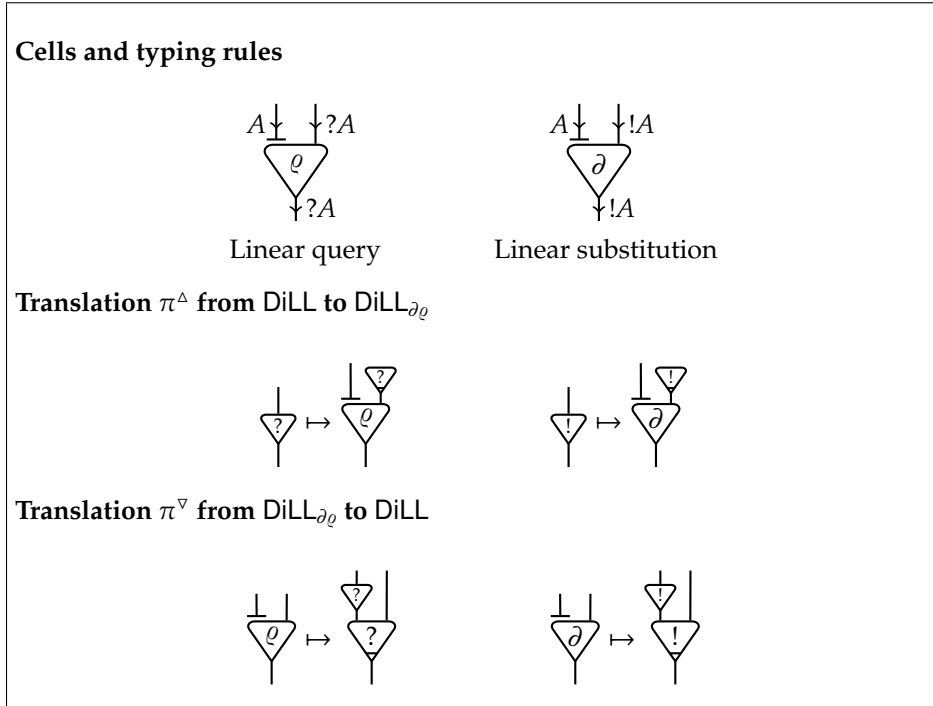


Figure 7.1: The cells ∂ and ϱ of DiLL $_{\partial\varrho}$, and the translation from and to DiLL. The cells are *not* commutative, but may be drawn with swapped inactive ports, hence the special marking of the left port. ϱ is switching, ∂ is not.

inside boxes in $\mathbf{b}_0(\pi')$, which therefore have each a corresponding polynet ν_i in $\mathbf{b}_0(\sigma)$ with $\mu_i \xrightarrow{-\text{err}^*} \nu_i$.

By inductive hypothesis there are standard chains R_i from μ_i to ν_i . Then we can go from π' to σ' by taking all reductions in the R_i s in order of depth (as reductions in different boxes can be safely reordered), and we conclude. \square

Notice that **ps**-equivalences and **pz**-reductions break this result, as these rely on the contents of a box, so they cannot be brought before reductions at greater depth in general.

7.2 DiLL $_{\partial\varrho}$

DiLL $_{\partial\varrho}$ is obtained by removing from DiLL the dereliction and codereliction cells and adding in the two cells ∂ (**linear substitution**) and ϱ (**linear query**) shown together with their lax typing rules in Figure 7.1. In the same figures are defined two translations from DiLL nets to DiLL $_{\partial\varrho}$ ones and back, defined by substituting the missing cells. We denote these translations by π^Δ and π^∇ respectively. DiLL $_{\partial\varrho}$ proof nets are defined by setting also ϱ as switching, as hinted by the corresponding translation into DiLL. It is in fact straightforward then that π in DiLL $_{\partial\varrho}$ is correct iff π^∇ is, and the same for σ^Δ .

The translation should already give an idea of what the missing reduction rules are. However we will introduce them already together with the labelling needed by Gandy's method.

7.2.1 Labelled Nets and Reduction

Given a polynet π , let $\text{fn}(\pi)$ be the set of occurrences of its nets. By the definitions given in Section 2.1.3, this gives for polynets with boxes a function $\text{fn}_!(\pi)$, the **flat nets** of π , giving the set of all nets occurring in π , whether at depth 0 or in a box. We define a **labelling** ℓ of a DiLL_{∂_0} π a function $\ell : \text{fn}_!(\pi) \rightarrow \mathbb{N} \setminus 0$, and a labelled DiLL_{∂_0} net σ as one having a labelling ℓ_σ .

Reduction rules then act also on the label of the unique net containing the redex at depth 0. More precisely, labelled contexts are defined in the same way as for regular polynets, but for the lack of a label for the net containing the hole at depth 0. Module plugging is defined as usual on the underlying nets, while the label of each simple module expands to be the label missing in the context. The precise definition goes as follows: for $\omega[\] = \psi[\delta[\] + \pi]$ where $\delta[\]$ is linear (so the only flat net of $\omega[\]$ missing a label), and $\mu = \sum \mu_i$ with μ_i simple modules, then

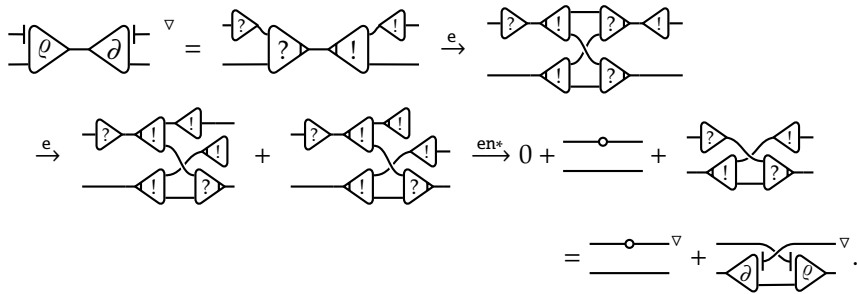
$$\ell^{\omega[\]}(\sigma) = \begin{cases} \ell^{\omega[\]}(\sigma) & \text{if } \sigma \in \text{fn}_!(\omega[\]) \setminus \{\delta[\]\}, \\ \ell^\mu(\sigma) & \text{if } \sigma \in \text{res}(\sigma') \text{ with } \sigma' \in \text{fn}_!(\mu) \setminus \{\mu\}, \\ \ell^{\mu_i}(\sigma) & \text{if } \sigma = \delta[\mu_i]. \end{cases}$$

Reductions become a function from labelled simple modules to labelled poly-modules. We will draw the label of a flat net by writing it inside a circle. Such reductions are shown in Figure 7.2, where also the markings of the variant $\text{DiLL}_{\partial_0}^\circ$ are shown, where “new” cuts are blocked (see Section 6.1). Accompanying the reductions is also the needed associative equivalence. A notable difference with DiLL will be however having confluence without the neutral reduction, which will be considered among the erasing reductions¹. Figure 7.3 shows the new pairs generating the associative equivalence, on top of the ones already known.

Lemma 7.8 (DiLL° simulates $\text{DiLL}_{\partial_0}^\circ$). *For π a $\text{DiLL}_{\partial_0}^\circ$ proofnet, if $\pi \xrightarrow{\circ} \sigma$, then $\pi^\nabla \xrightarrow{\text{en}^+} \sigma^\nabla$ in DiLL° , where in both $\sim = \overset{\circ}{\sim}$.*

Proof. It suffices to translate all redexes (and conversion instances) and reduce (resp. convert) them. We show only the linear substitution on linear query and the linear substitution on box. The others are straightforward.

Regarding the former, we have



¹In fact, considering neutral reduction would even introduce some complications. As it stands, neutral reduction breaks local coherence in DiLL_{∂_0} .

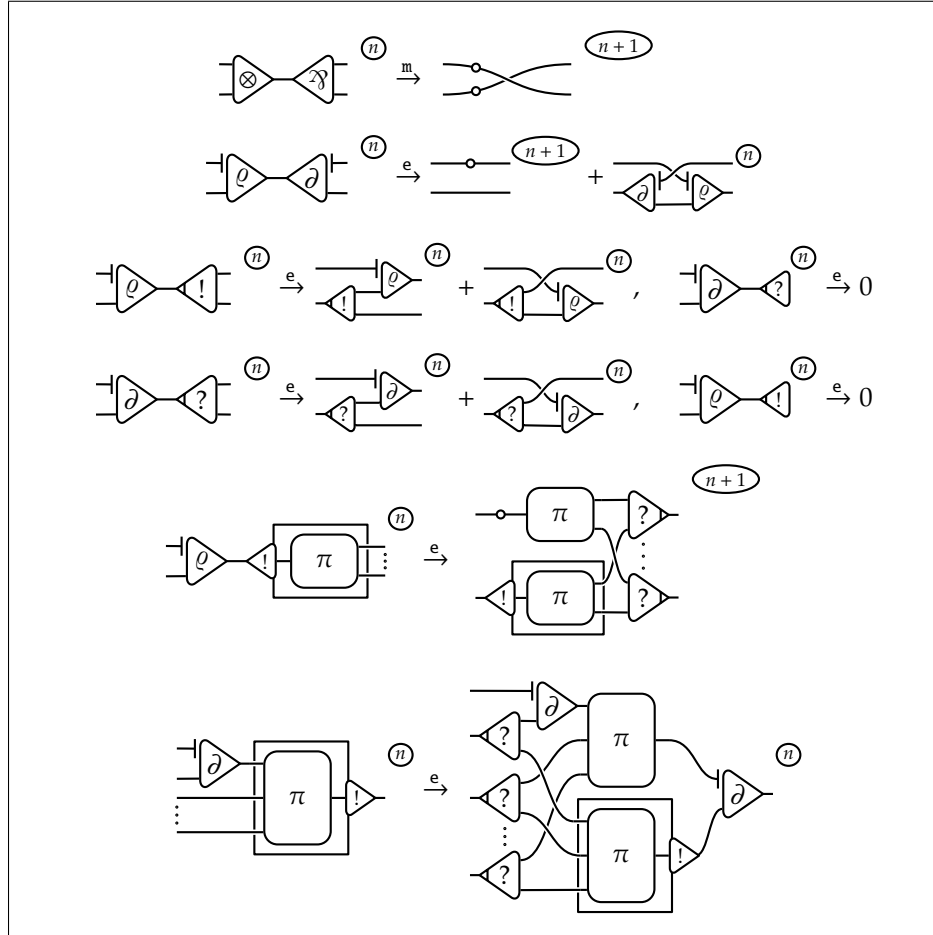


Figure 7.2: Reduction rules of labeled DiLL_{∂ϑ}[∘]. All combinations not shown are the same as in DiLL, with labels left unchanged.

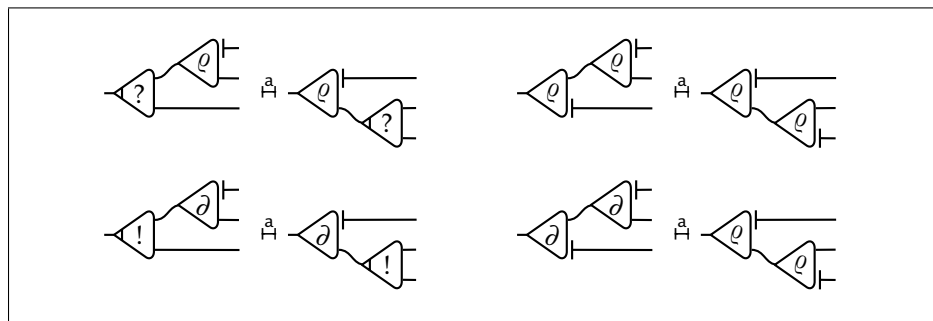
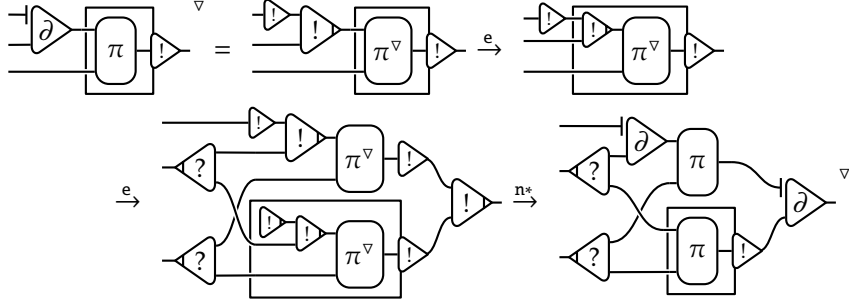


Figure 7.3: The new associative equivalences.

As for the latter, we have (with a box with only two auxiliary ports for brevity)



□

A direct consequence of the above lemma and of Theorem 6.1 is the finiteness of developments of (unlabelled as well as labelled) $\text{DiLL}_{\partial\varrho}$.

Proposition 7.9. *The reduction me on labelled $\text{DiLL}_{\partial\varrho}^\circ$ proofnets is strongly normalizing modulo \approx .*

Lemma 7.10 ($\text{DiLL}_{\partial\varrho}$ simulates DiLL). *If in DiLL $\pi \xrightarrow{e} \sigma$ then $\pi^\Delta \xrightarrow{\text{en}^+} \sigma^\Delta$ in $\text{DiLL}_{\partial\varrho}$.*

Proof. Not much different from the proof of Lemma 7.8. □

7.2.2 Confluence of Labelled Non Erasing Reduction

Having in hand Proposition 7.9, we aim at proving confluence of the labelled and non erasing reduction in $\text{DiLL}_{\partial\varrho}$, by refollowing the steps taken for proving confluence of DiLL . In $\text{DiLL}_{\partial\varrho}$ the erasing and non erasing reductions have the same definition as in DiLL , just replacing the roles of dereliction and codereliction with the new cells.

The rest of this section will be dedicated to prove the following result. As usual, \sim denotes either \approx or $\overset{\text{asp}}{\sim}$.

Lemma 7.11 (Confluence of $\neg\text{er}$). *The non erasing reduction is $\text{CR}\sim$ for ℓ - $\text{DiLL}_{\partial\varrho}$.*

As we have Proposition 7.9 (finite developments), and lemmas 6.10, 6.11 and 6.12 are trivially still valid in $\text{DiLL}_{\partial\varrho}^\circ$, again it suffices to check the two following results in $\text{DiLL}_{\partial\varrho}^\circ$.

Lemma 7.12 (Local confluence of $\neg\text{er}$). *The non erasing reduction is locally confluent modulo \sim on $\text{DiLL}_{\partial\varrho}^\circ$.*

Proof. First let us note that we cannot use the simulation result (Lemma 7.8), as it, together with $\text{CR}\sim$ of DiLL° , gives indeed $\text{CR}\sim$ in $\text{DiLL}_{\partial\varrho}^\circ$, but of plain e reduction (simulating DiLL reduction uses erasing reductions). Apart the confluence diagrams in common with DiLL (which can be seen to using only non erasing reductions if the critical peak is non erasing), the only non trivial diagrams are those of two ∂ cells against a box and ∂ on box on ϱ .

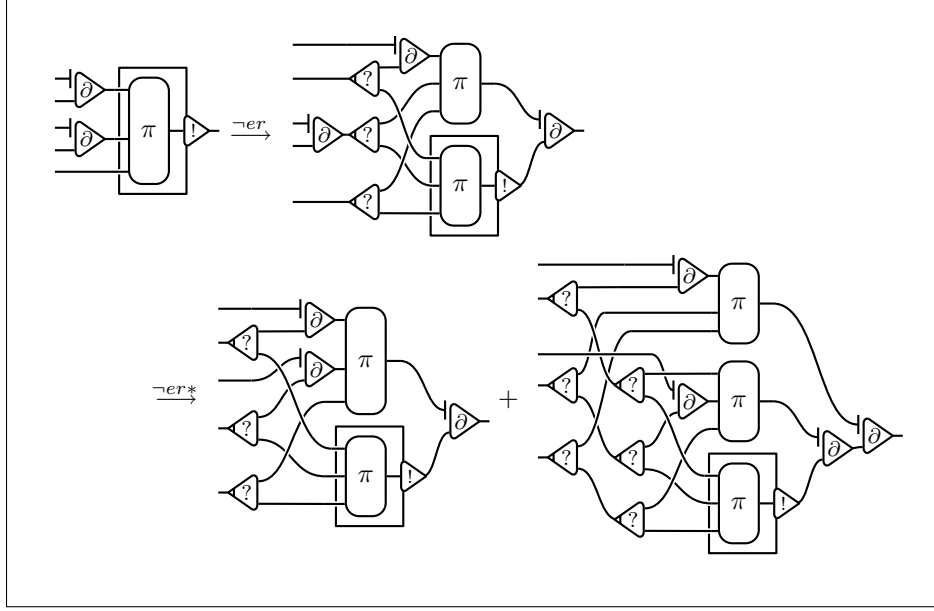


Figure 7.4: Reduction of the critical peak with two ∂ cells on a box. The box has only one other auxiliary port for brevity. Starting with the other ∂ cell is completely symmetric, and an **a**-conversion equates the two results. Labels are omitted as they are unaffected.

For the former, Figure 7.4 shows the reduction starting with one of the two ∂ cells. The other ∂ cell leads to a symmetric form where the only difference is the order of the rightmost ∂ cells and the shape of the contraction trees, so that an **a**-conversion equates the two results. We do not write the labels as they remain unchanged throughout the reduction.

For the latter, Figure 7.5 shows the confluence diagram. We recall that $\xrightarrow{\Sigma \mathbf{r}}$ is the reduction that make at most a step in each simple addend. \square

Lemma 7.13 (Local coherence of \vdash with $\neg er$). *The relation \vdash generating \sim is locally coherent with non erasing reduction.*

Proof. Again, we must check all coherence diagrams. The majority is trivial, or a variant over the ones we have shown for local confluence. We show an interesting one in Figure 7.6, a ∂ cell on a **s**-convertible box. This is the only diagram that needs the “mixed” associativity rule between cocontraction and linear substitution. Labels are omitted as they are unaffected. Recall that $\pi_1, \pi_2 \neq 0$ is required by the **s**-conversion, so that no step is forbidden by non erasing discipline. \square

7.3 Proof of the Theorem

In the first part of this section, we will briefly finalize the proof of the conservation theorem for DiLL_{∂_e} , while the second will show how to transport the result to DiLL .

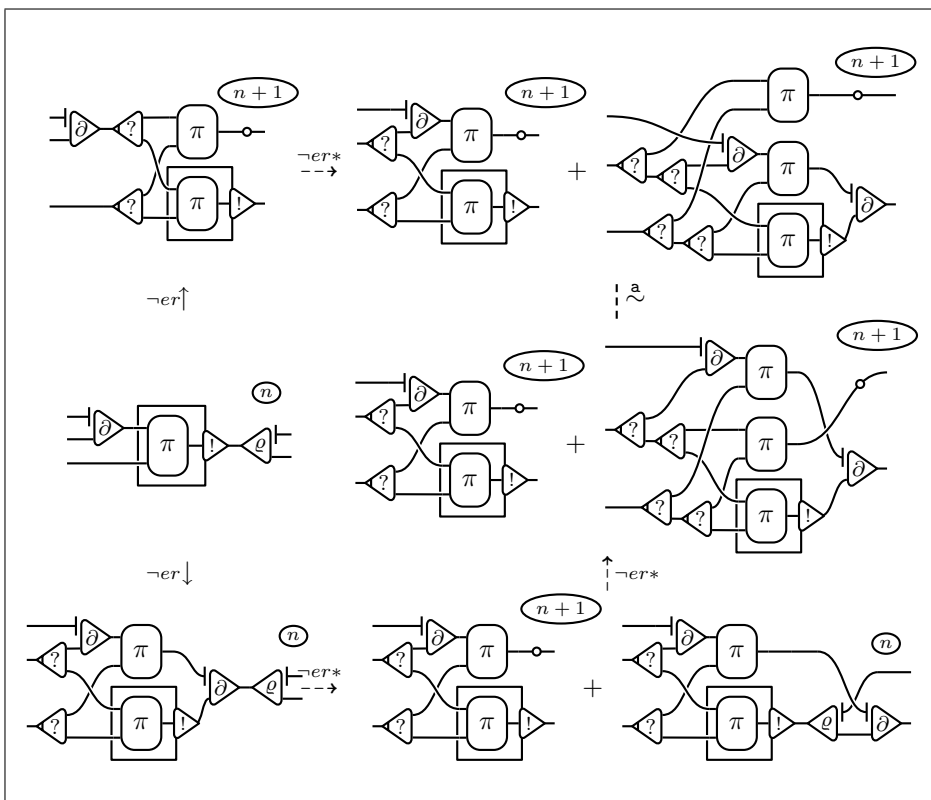


Figure 7.5: Confluence diagram of the ∂ on box on ρ critical peak.

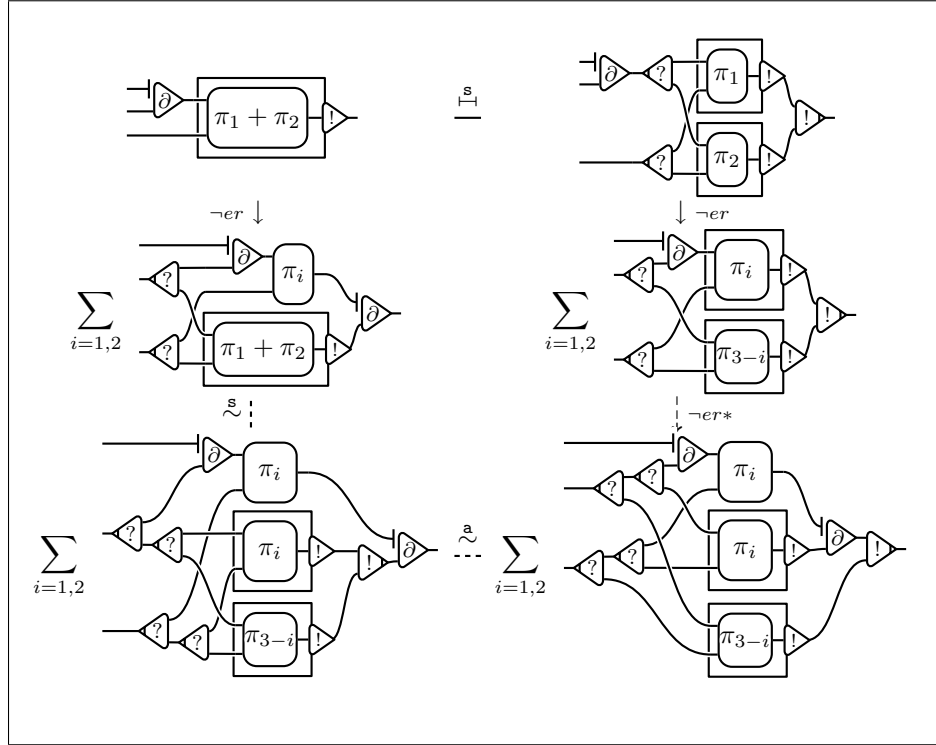


Figure 7.6: Coherence diagram of a ∂ cell on an s -conversion.

For the remainder of this section we make a slight change to the reduction rules: we will consider *only* total pull reductions, leaving out the partial one. This *breaks* the Church-Rosser property, but we have the following easy property.

Lemma 7.14. *If π is strongly normalizing for all reductions without the partial pull one, then it is so for all reductions.*

Proof. Straightforward, as the partial p -reduction can be recovered as an s -conversion followed by a total p -reduction and an n one. \square

This change is required for Lemma 7.18.

7.3.1 Conservation for DiLL_{∂_Q}

Proposition 7.15. *Given a net π in DiLL_{∂_Q} , we have that $\pi \in \text{SN} \sim \iff \pi \in \text{WN}_{\neg er}$.*

In order to arrive to this result we employ as already explained Gandy's method. We thus first prove the following intermediate result. The proof of the proposition is postponed to the end of this section.

Lemma 7.16. *Given a net π in DiLL_{∂_Q} , if $\pi \in \text{WN}_{\neg er}$ then π is $\text{SN} \sim$ for $\neg er$.*

Proof. We define a measure $\llbracket \pi \rrbracket_G \in \mathbb{N}$ on labelled DiLL_{∂_Q} nets, and show it is

increasing on reductions and invariant on conversions. Let

$$\begin{aligned} \ell_1(\pi) &:= \sum_{\lambda \in \text{fn}(\pi)} \ell_\pi(\lambda), & \#\partial(\pi) &:= \#\{c \in \mathbf{c}_!(\pi) \mid \sigma(c) = \partial\}, \\ \#_i^!(\pi) &:= \#\{c \in \mathbf{c}_i(\pi) \mid \sigma(c) = !_2\} + \sum_{\substack{B \in \mathbf{b}_i(\pi) \\ \sigma(B) \neq 0}} (2\#\sigma(B) - 1). \end{aligned} \quad (7.1)$$

The parameter i lets us slice $\#^!$ by depth. We then let

$$\llbracket \pi \rrbracket_G := \ell_1(\pi) + \#\partial(\pi) + \sum_{i=0}^{d(\pi)} (1+i) \#_i^!$$

Put in words, we sum all labels, and then the number of $!$ cells (cocontractions $!_2$ and boxes), weighted by their depth and, in the case of boxes, by $2n-1$ where n is the number of addends in them.

Suppose $\omega[]$ has its hole at depth d , and let λ be a simple module. We clearly have

$$\begin{aligned} \ell_1(\omega[\lambda]) &= \ell_1(\omega[]) + \ell_1(\lambda), \\ \#\partial(\omega[\lambda]) &= \#\partial(\omega[]) + \#\partial(\lambda), \\ \#_i^!(\omega[\lambda]) &= \begin{cases} \#_i^!(\omega[]) & \text{if } i < d, \\ \#_i^!(\omega[]) + \#_{i-d}^!(\lambda) & \text{otherwise.} \end{cases} \end{aligned}$$

In case λ is a non zero polymodule then the last two of the above equalities become inequalities \geq , because of the possible additive duplication of the context and also the possible increase of the number of addends in a box. This shows us that we can always check each of the measures defined in (7.1) on the redex-reduct pairs alone. If all increase and at least one does so strictly then $\llbracket \cdot \rrbracket_G$ increases strictly.

Let therefore $\omega[\lambda] \xrightarrow{\text{red}} \omega[\sum_{i=1}^k \mu_i]$, with λ the redex fired. Let us see all the cases.

- Multiplicative reduction: only ℓ_1 changes, and $\ell_1(\mu_1) = \ell_1(\lambda) + 1 > \ell_1(\lambda)$.
- ∂ on ϱ : we have $\#\partial(\mu_1 + \mu_2) = 1 = \#\partial(\lambda)$, $\#_i^!(\mu_1 + \mu_2) = 0 = \#_i^!(\lambda)$, and $\ell_1(\mu_1 + \mu_2) = 2\ell_1(\lambda) + 1 > \ell_1(\lambda)$.
- ϱ on cocontraction: we have $\#\partial(\mu_1 + \mu_2) = 0 = \#\partial(\lambda)$, $\#_0^!(\mu_1 + \mu_2) = 2 > 1 = \#_0^!(\lambda)$, the other $\#_i^!$ are zero, and ℓ_1 is unaffected.
- ∂ on contraction: as above, but it is the number of ∂ cells that increases.
- ϱ on a box B : we have $k = \#\sigma(B) \geq 1$. We then have $\#\partial(\sum_i \mu_i) \geq \#\partial(\mu_1) \geq \#\partial(\sigma(B)) = \#\partial(\lambda)$, $\#_0^!(\sum_i \mu_i) \geq \#_0^!(\mu_1) \geq 2k - 1 = \#_0^!(\lambda)$, while for $i > 0$ we have $\#_i^!(\sum_i \mu_i) \geq \#_i^!(\mu_1) \geq \#_{i-1}^!(\sigma(B)) = \#_i^!(\lambda)$, and finally $\ell_1(\sum_i \mu_i) \geq \ell_1(\mu_1) \geq 1 + \ell_1(\lambda) + \ell_1(\sigma(B)) > \ell_1(\lambda)$.
- ∂ on a box B : as above nothing can decrease, while the total number of ∂ cells increases.

- Box or cocontraction on box: labels are unchanged, but the contribution of the cell entering the box increases as its depth does. Note that it is true because the box must be non 0.
- Box or cocontraction on contraction: labels do not decrease (in fact in case of box on contraction, they do increase by duplication), while the contribution of the ! cell is redoubled.

Next, we can see it is invariant on conversions.

- Associative equivalence: labels are constant, while ! cells are clearly unaffected when moving them around at the same depth.
- Push equivalence: contractions do not play any role in the measure.
- Bang sum equivalence: suppose we have a box B containing $\sigma + \pi$ (both non zero) which is being splitted into two boxes B_1 and B_2 , and suppose this happens at depth d . The context is clearly unchanged, and the flat nets are exactly the same. Then the only difference can be equated by the following chain of equalities (the $1 + d$ at the start is the contribution of the cocontraction joining boxes B_1 and B_2):

$$\begin{aligned} (1 + d) + (1 + d)(2 \# \sigma(B_1) - 1) + (1 + d)(2 \# \sigma(B_2) - 1) \\ = (1 + d)(2(\# \pi + \# \sigma) - 1) = (1 + d)(2 \# \sigma(B) - 1). \end{aligned}$$

We can therefore conclude by employing Lemma 7.5. \square

Lemma 7.17. *In DiLL_{∂_e} if $\pi \xrightarrow{er*} \pi'$ then $\pi \sim^{er*} \pi'$.*

Proof. In other words, we must prove that the erasing reduction can be delayed with respect to the conversions \vdash . First thing, we can reason separately for canonical and non canonical reductions. In fact if $\xrightarrow{1*} \subseteq \sim^{1*}$ and $\xrightarrow{2*} \subseteq \sim^{2*}$ for two reductions then $\xrightarrow{12*} \subseteq \sim^{12*}$, by shoving right each consecutive group of 1 or 2-reductions.

For erasing non canonical reductions for the conversion instance to be present after erasure it must be the case that the two are orthogonal, so $\rightarrow \vdash \subseteq \vdash \rightarrow$ and the result is straightforward.

Now take $\pi \xrightarrow{c*} \pi'$. By direct inspection of the rules one can see that $\xrightarrow{c} \vdash \subseteq (\vdash)^k \xrightarrow{c+}$, where $k \leq 2$. The particular cases arising are:

- when a neutral redex “blocks” a conversion instance: it is there that $k = 2$, as an additional conversion is needed for the additional (co)contraction;
- a pull redex is then object of an **s**-conversion: the diagram is closed by a first **s**-conversion followed by two **p**-reductions and an **n** one.

We will use this inclusion as a rewriting of the sequence taking π to π' . Let $a_k \cdots a_1 a_0$ be a sequence of **c**-reductions and single-step conversions from π to π' . Define by induction the following function:

$$i(0) := 0, \quad i(h+1) := \begin{cases} i(h) + 1 & \text{if } a_h = \vdash, \\ 2i(h) & \text{if } a_h = \xrightarrow{c}. \end{cases}$$

Now consider

$$|a_k \cdots a_0| := \left(\#_c(\pi) - \# \{ h \mid a_h = \overset{c}{\rightarrow} \}, \sum_{a_h = \overset{c}{\rightarrow}} i(h) \right)$$

where $\#_c$ is the measure defined in the proof of Proposition 6.8. The number $\#_c$ maximizes the number of consecutive c -reductions, as c -reductions cannot leave DiLL° . The first component of $|\vec{a}_h|$ is therefore always positive.

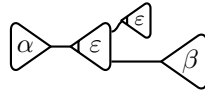
Now, $|\vec{a}_h|$ decreases for lexicographic ordering for each rewriting of $\overset{c}{\rightarrow}$ into the corresponding $(\dashv)_k \overset{c}{\rightarrow}$. Suppose such rewriting happens at position h from the end of the sequence. If the first component of $|\cdot|$ remains constant we have that the contribution of the second component $(\dashv)_k \overset{c}{\rightarrow}$ is $i(h)$ in place of $i(h)+1$. For the previous elements of the sequence, if $k=1$ all $i(h')$ decrease. If $k=2$ all the elements are shifted by one, but as *after* rewriting $i(h+2) = 2i(h) + 2$ just as $i(h+1)$ was *before* it, all the contributions are unchanged. \square

Lemma 7.18. *In $\text{DiLL}_{\partial\varrho}$ if $\pi \xrightarrow[\sim]{er*} \xrightarrow[\sim]{\neg er} \pi'$ then $\pi \xrightarrow[\sim]{\neg er} \xrightarrow[\sim]{*} \pi'$.*

Proof. First by the above lemma we can transform the reduction chain into $\pi \xrightarrow[\sim]{er*} \xrightarrow[\sim]{\neg er} \pi'$. Then we proceed by induction on the length of the initial chain of erasing reductions. The base step is trivial, and the inductive one boils down to showing $\xrightarrow[\sim]{er} \xrightarrow[\sim]{\neg er} \subseteq \xrightarrow[\sim]{\neg er} \xrightarrow[\sim]{er*}$.

For all erasing reductions but the \mathbf{n} one, the result is achieved by noticing that the erasing step cannot have created the non erasing redex fired immediately afterwards. Most of the times, the two reductions are orthogonal and therefore commute easily. This does not happen only when the erasing step is one on an auxiliary port of a box (a box entering step) and the non erasing one is on the box itself. All cases for the non erasing step (dereliction, contraction, codereliction and cocontraction) are then easily handled. Partial pull reduction would break the result, so we have explicitly left it out.

The \mathbf{n} reduction is peculiar as it can “create” the wire where the following non erasing reduction takes place. By lax typing however, one of the cells taking place in the non erasing step can also interact with the (co)contraction of the \mathbf{n} -reduction. The situation is the following



where ε is either $?$ or $!$ and α can interact both with the (co)contraction and with β . Then

- if α is duplicated by ε_2 (i.e. α is a box, cocontraction and contraction): one copy can then reduce by a non erasing step with β , the other will be erased by ε_0 and \mathbf{n} -reductions will close the diagram;
- if α is a (co)dereliction, the reduction against ε_2 generates a sum: in one addend we can follow with the non erasing α on β , then close by erasing the other addend by ε_0 on (co)dereliction and the weakening-coweakening pair in the addend left.

- if α is a box and ε_2 a cocontraction, then we can make ε_2 enter α , fire the α on β redex, and then make all the residuals of ε_0 n -reduce with all the residuals of ε_2 . \square

Lemma 7.19. *In $\text{DiLL}_{\partial\varrho}$ erasing reduction is $\text{SN}\sim$.*

Proof. Erasing reduction can all be carried out in $\text{DiLL}_{\partial\varrho}^\circ$, so Proposition 7.9 gives the desired result. \square

Proof of Proposition 7.15. Clearly only the part $\pi \in \text{WN}_{-er} \implies \pi \in \text{SN}\sim$ is to be shown.

Let $\pi \in \text{WN}_{-er}$, which by Lemma 7.16 means that $\pi \in \text{SN}\sim$ for non erasing reduction. Suppose we have an infinite chain R of mec -reductions interleaved by \sim -conversions starting from π . If we are able to build by induction an infinite chain of non erasing reductions starting from π we can conclude by contradiction.

Let R_0 be the empty chain of reductions. Now suppose we have a reduction R_k of the shape $\pi \xrightarrow[\sim]{(-er)^k} \pi_k$ ($\pi_0 := \pi$) and an infinite reduction R' starting from π_k . By Lemma 7.19 R' cannot be solely made of erasing steps, so it must necessarily start with a segment of the shape $\pi_k \xrightarrow[\sim]{er^*} \xrightarrow[\sim]{-er} \pi'$, with an infinite reduction starting from π' . Now Lemma 7.17 ensures that we have $\pi \xrightarrow[\sim]{(-er)^{k+1}} \pi_{k+1}$ with $\pi_{k+1} \xrightarrow{*} \pi'$, so that π_{k+1} also has an infinite reduction. \square

7.3.2 Transferring Conservation from $\text{DiLL}_{\partial\varrho}$ to DiLL

Objective of this section is to use $\text{DiLL}_{\partial\varrho}$ to finally prove Theorem 7.2.

Dead ends. We will call **dead end** any DiLL net λ with one free port such that for any linear context $\omega[\]$ which is normal for non erasing reduction, also $\omega[\lambda]$ is normal for $\xrightarrow{-er}$.

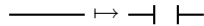
Remark 7.20. In fact, dead ends can be characterized as those nets that are normal for $\xrightarrow{-er}$ and the port above the only free one is either inactive, or the principal port of a weakening, coweakening or 0-box.

We will denote *any* dead end by a vertical bar interrupting a wire:



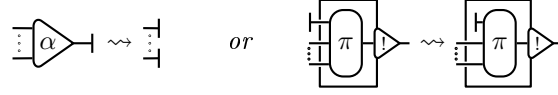
The simplest examples of dead ends are weakenings and coweakenings.

Lemma 7.21. *For any $\text{DiLL}_{\partial\varrho}$ proof net π , if π is weakly normalizable for non erasing rules, then so is in $\text{DiLL}_{\partial\varrho}$ a polynet π' obtained from π by interrupting certain wires by any two dead ends:*



Proof. Because of Proposition 7.15, we just need to prove that $\pi \in \text{SN}\sim$ implies $\pi' \in \text{SN}\sim$ by well founded induction on π . In fact if we take any σ' with $\pi' \rightrightarrows \sigma'$ then the redex fired must be in π also. Thus, there is σ with $\pi \rightrightarrows \sigma$ where σ' is obtained by interrupting some wires of σ , so that $\sigma' \in \text{SN}\sim$ by inductive hypothesis. We conclude, as therefore π' is in $\text{SN}\sim$ also. \square

Lemma 7.22. *Suppose π' is a $\text{DiLL}_{\partial\varrho}$ polynet obtained from a polynet π by performing a number of subsequent substitutions of the shape*



where α is any unicell symbol, i.e. any symbol but a box. Then $\pi' \in \text{WN}_{\neg er} \implies \pi \in \text{WN}_{\neg er}$. In particular, the hypothesis apply if $\pi \xrightarrow{*} \pi'$ by reductions of weakening and coweakening excluding the weakening on box one.

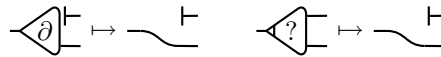
Proof. Let us reason by induction on the normalizing reduction of π' . If π' is normal then clearly so is π . Otherwise, the redex fired in $\pi' \xrightarrow{\neg er} \sigma'$ must also be in π . Once such redex is fired and we get $\pi \xrightarrow{\neg er} \sigma$ we must show that σ' can be obtained from σ with the above rules. The box of (various instances of) the right rule might have been

- duplicated by a contraction: the dead end is then found in front of a contraction, which can be “eaten” by the left rule, and by making the two resulting dead ends enter their boxes with the second one;
- duplicated linearly by a ∂ or ϱ : as above, but one dead end need not enter the box.

Apart from that, one may need to repeat both rules several times to account for any kind of duplication. \square

The \triangleleft order. Given polynets π and σ in $\text{DiLL}_{\partial\varrho}$, we define $\sigma \triangleleft \pi$ when each $\lambda \in \sigma$ is obtained from a net $\mu \in \pi$ by

1. substituting *some* of the ∂ cells or contractions at depth 0 in μ in the following way, where the lax type of the new wire is obtained as usual by the meet of the lax types of the melded wires:



(notice that commutativity gives two choices for the substitution of a contraction);

2. taking then a subnet *not containing* any of the dead ends introduced in the previous point, and *not introducing* any new free ports.

We see that \triangleleft is closed by affine contexts. Also \triangleleft is a preorder, as it is transitive and reflexive. Moreover it is an order, as antisymmetry holds. Indeed, if $\pi \triangleleft \sigma \triangleleft \pi$ then no substitution can be done in step 1 as they make the number of cells strictly decrease. However we will not make use of this ordering structure.

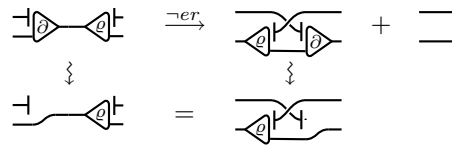
Lemma 7.23. *If $\sigma \triangleleft \pi$ and $\pi \xrightarrow{\neg er} \pi'$, then there is $\sigma' \triangleleft \pi'$ such that $\sigma \xrightarrow{\neg er*} \sigma'$.*

Proof. Take any $\lambda \triangleleft \pi$ simple, we show there is $\lambda' \triangleleft \pi'$ with $\lambda \xrightarrow{\neg er*} \lambda'$ by cases on the reduction. If it does not touch any cell substituted with a dead end, then it is straightforward. Notice that if the redex is preserved through the point 1 of the construction, then either it is entirely preserved or entirely deleted by

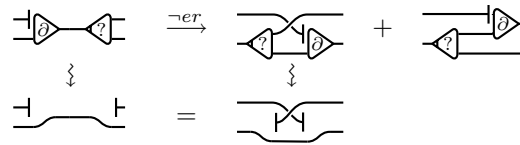
point 2 because of the condition on free ports. In the latter case there will be no reduction in λ .

Now take any non erasing redex μ containing at least a ∂ or a contraction changed in point 1, and let μ' be its reduction. It suffices then to see that if $\nu \triangleleft \mu$ and ν is normal (because the ∂ or contraction cell generating the redex has been substituted), then $\nu \triangleleft \mu'$ also. We show the cases, where we indicate how substitution can be carried out to gain the result:

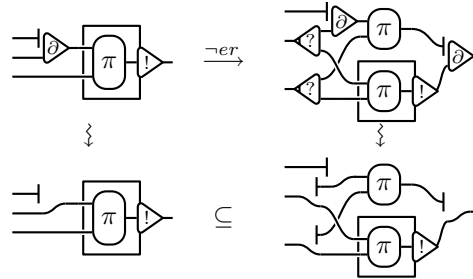
- ∂ on ϱ :



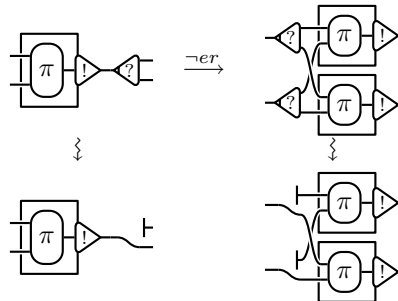
- ∂ on contraction: here we have 3 combinations for an actual substitution (∂ , contraction or both); the first two are the same as above, the third is the following:



- ∂ on box:



- contraction on box:

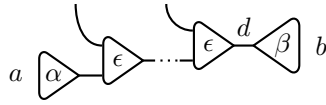


as any net obtained from the right one cannot contain any of the dead ends (and cannot introduce new free ports), then it cannot contain the whole box; it can then be obtained also from the right. \square

The following is the central point where lax typing makes a true difference.

Lemma 7.24. *If $\sigma \triangleleft \pi$ and π is normal for non erasing reduction, then so is σ .*

Proof. Suppose $\sigma \ni \lambda \triangleleft \mu \in \pi$ with λ not normal, and take the two cells a and b (none of which can be a weakening, coweakening or 0-box) which are active in it and c the cut wire connecting them (non erasing reduction is not a generalized one). The wire c cannot be \mathcal{U} -typed. In μ , which is normal, the wire c must be thus interrupted by cells that got substituted in point 1. As c is not \mathcal{U} -typed, so must be all its segments in μ . Furtherly, as none of its segments in μ can therefore be a cut, it means that along it there are either only contractions, or only ∂ cells, and all in the same direction. So in μ we have, without loss of generality:

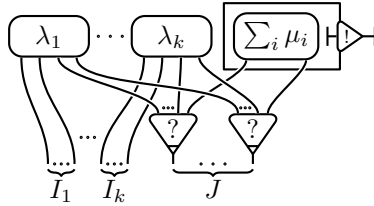


where $\epsilon = ?_2$ or ∂ . Now d must necessarily be a non \mathcal{U} -typed cut. As b cannot be part of an erasing redex, d is a non erasing reducible cut which is a contradiction. \square

Lemma 7.25. *Take*

- *a polymodule π in DiLL_{∂_0} , with interface divided into $I_\pi = I_1 + \dots + I_k + J$ (the linear ones and the exponential one) with $k \geq 1$, so that all the wire above J are laxly typed by $?$ or \mathcal{U} ,*
- *and a corresponding linear context $\omega[\]$ where all I_i are not linked neither with I_j for $j \neq i$ nor with J^2 ,*

such that $\omega[\pi]$ is $\text{SN}\sim$ for non erasing reduction. Then take any net σ of the shape



where $\lambda_i \triangleleft \pi$ and $\mu_i \triangleleft \pi$ for all i , the n -ary contractions stand for any tree of contractions with n leaves, and the wires above J are laxly typed like the corresponding ones in π .

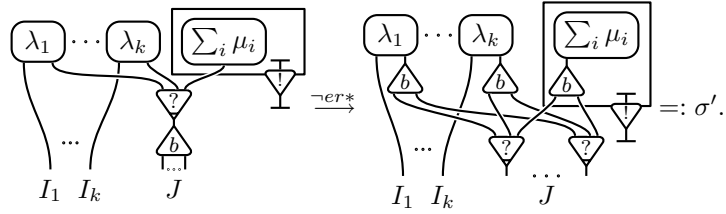
We then have that $\omega[\sigma] \in \text{WN}_{-er}$.

Proof. We proceed by well founded induction on $\omega[\pi]$, with a secondary induction on the size of $\omega[\]$.

Using secondary induction, we reduce to a case where no wire between $\omega[\]$ and σ is a non erasing reducible cut in $\omega[\sigma]$. Suppose there is such a cut c between cells a and b , we show how to build π' , $\omega'[\]$, λ'_i and μ'_i so that $\omega'[\pi'] = \omega[\pi]$ and $\omega'[\]$ is smaller than $\omega[\]$. Several cases are possible.

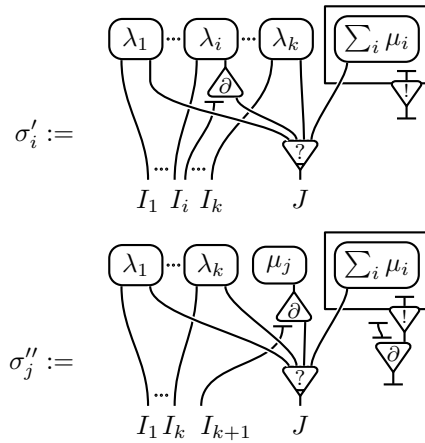
²We are implicitly identifying I_π with the hole I_ω^\square .

- c traverses I_i , at least one among a and b is in ω (suppose a): π' and λ'_i are then obtained by glueing a to π and λ_i respectively, adding all the other ports of a to I_i , and leaving all other λ_j and μ_j unchanged; $\omega'[\]$ is $\omega[\]$ without b . We have that $\lambda_j \triangleleft \pi'$ for $j \neq i$ as c is not in λ_j , and $\lambda'_i \triangleleft \pi'$ trivially. Moreover any “illegal” switching path linking the hole of $\omega'[\]$ would either be one also for $\omega[\]$, or could be completed by passing through a (active ports are always reachable by switching paths ending on other ports of the same cell). By secondary inductive hypothesis, $\omega'[\sigma'] = \omega[\sigma]$ is WN_{-er} .
- c traverses I_i , none of a and b are in $\omega[\]$: by the constraint on switching paths c must be between two ports in I_i , so we can safely move c inside π and λ_i without breaking the shape of σ .
- c traverses J , and a is one of the contractions of σ on J . b must be in $\omega[\]$, otherwise the condition on paths would be broken. We have two subcases.
 - If b is a duplicable cell, i.e. a cocontraction or a box, then (simplifying a bit the picture for the sake of clarity) glueing b to σ we obtain



If $\omega'[\]$ is $\omega[\]$ deprived of b , then $\omega[\sigma] \xrightarrow{-er*} \omega'[\sigma']$. Glueing b to σ then gives the secondary inductive hypothesis which in turns offers $\omega[\sigma'] \in \text{WN}_{-er}$.

- If b is a ∂ cell, by setting $\omega'[\]$ as before and performing reduction we get $\omega[\sigma] \xrightarrow{-er*} \sum_i \omega[\sigma'_i] + \sum_j \omega[\sigma''_j]$ where



For each i , glue b to π' by moving its linear port to I_i (notice this cannot break the path condition): we have then that σ'_i falls into the hypotheses. Indeed, for $h \neq i$ we have $\lambda_h \triangleleft \pi'$ by breaking b

with step 1 of the definition. Then, for each σ_j'' also falls into the hypotheses by the same reasoning (notice the ∂ underneath the box is a dead end by definition), where however the linear wire of b is moved to a new linear interface I_{k+1} .

The absence of cuts between $\omega[\]$ and σ implies by lax typing constraints that there cannot be any reducible cut between $\omega[\]$ and π also. We have thus three cases: either $\omega[\pi]$ is normal for non erasing reduction, or it has a non erasing redex entirely in $\omega[\]$, or it has one in π . In the first case, Lemma 7.24 assures us that $\omega[\sigma]$ is also normal.

In the second one, let us reduce the redex in $\omega[\]$ in all addends of $\omega[\pi]$, getting $\omega[\pi] \xrightarrow{\neg er^+} \omega'[\pi]$. By primary induction this gives $\omega'[\sigma] \in \text{WN}_{\neg er}$. As $\omega[\sigma] \xrightarrow{\neg er} \omega'[\sigma]$ we are done.

In the last case with $\pi \xrightarrow{\neg er} \pi'$, Lemma 7.23 assures the existence of λ'_i with $\lambda_i \xrightarrow{\neg er^*} \lambda'_i$, so that we get σ' with $\omega[\sigma] \xrightarrow{\neg er^*} \omega[\sigma']$ which by induction is $\text{WN}_{\neg er}$. \square

Main lemma and proof. We can finally arrive to the main lemma opening the way for the central theorem.

Lemma 7.26. *Let π be a pure DiLL proofnet. If $\pi \in \text{WN}_{\neg er}$ then $\pi^\Delta \in \text{WN}_{\neg er}$ in $\text{DiLL}_{\partial \varrho}$.*

Proof. First, using Theorem 7.1, consider a standard normalizing reduction of π to normal form. Let us reason by induction on the length of such reduction, by cases on the first reduction. If π is normal for $\xrightarrow{\neg er}$, then so is π^Δ .

First of all, we can consider without loss of generality that the first reduction is at depth 0. Indeed, suppose that the first reduction of the chain R is at depth $i > 0$. As R is standard, it means that there is no non erasing redex with depth lesser than i in π , and therefore in π^Δ also. Now consider the sum $\sigma := \mu_1 + \dots + \mu_k$ where $\{\mu_i\} = \wp_i(\pi)$, i.e. all the nets at depth i of π . Free ports might not match, but we can just ignore it, as it is just a convenient way to consider all reductions on such nets together³. All R takes place in σ and normalizes it, now starting from depth 0. Once the result is proved for σ , we have $\sigma^\Delta = \mu_1^\Delta + \dots + \mu_k^\Delta$ is WN for non erasing reduction. Replugging all nets back into their place in π^Δ , we obtain that it is $\text{WN}_{\neg er}$ also.

Suppose therefore that $\pi \xrightarrow{\neg er} \sigma \in \text{WN}_{\neg er}$, with the reduction at depth 0. For every reduction which does not touch derelictions or coderelictions, the inductive step is immediate, as the translation in $\text{DiLL}_{\partial \varrho}$ leaves the redex untouched, so that $\pi^\Delta \xrightarrow{\neg er} \sigma^\Delta$. Let us see all the remaining cases. Let $\pi = \omega[\lambda]$ with λ the redex fired, and μ its reduct. We can suppose that $\omega[\]$ is linear, as closing by sum is trivial.

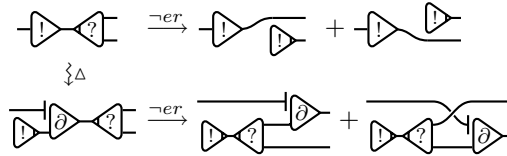
Dereliction on codereliction.

$$\begin{array}{c}
 \begin{array}{ccc}
 \begin{array}{|c|} \hline \text{!} \\ \hline \end{array} & \begin{array}{|c|} \hline \text{?} \\ \hline \end{array} \\
 \hline
 \end{array} & \xrightarrow{\neg er} & \text{---} \\
 \downarrow \Delta \\
 \begin{array}{ccc}
 \begin{array}{|c|} \hline \text{!} \\ \hline \end{array} & \begin{array}{|c|} \hline \partial \\ \hline \end{array} & \begin{array}{|c|} \hline \text{e} \\ \hline \end{array} & \begin{array}{|c|} \hline \text{?} \\ \hline \end{array} \\
 \hline
 \end{array} & \xrightarrow{\neg er} & \begin{array}{|c|} \hline \text{!} \\ \hline \end{array} \begin{array}{|c|} \hline \text{?} \\ \hline \end{array} + \begin{array}{|c|} \hline \text{!} \\ \hline \end{array} \begin{array}{|c|} \hline \partial \\ \hline \end{array} \begin{array}{|c|} \hline \text{?} \\ \hline \end{array} \begin{array}{|c|} \hline \text{?} \\ \hline \end{array} \rightsquigarrow \text{---} + \text{!H!} =: \mu'_1 + \mu'_2
 \end{array}$$

³We can in fact add in the needed conclusions with some dead ends.

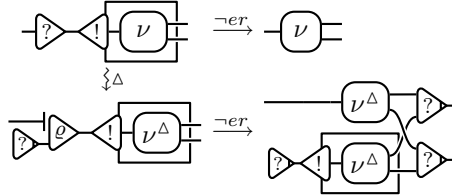
The last passage is backed up by Lemma 7.22. Now $\omega^\Delta[\mu^\Delta] \in \text{WN}_{\neg er}$ implies that both $\omega^\Delta[\mu'_1] = \omega^\Delta[\mu^\Delta]$ and $\omega^\Delta[\mu'_2]$ weakly normalizable, the latter by using Lemma 7.21.

(Co)dereliction on co(co)ntraction

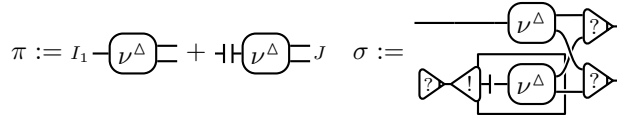


A direct application of Lemma 7.22 gives the result, as $\omega^\Delta[\lambda^\Delta] \xrightarrow{\neg er, *} \omega^\Delta[\mu^\Delta]$ with the latter being (co)weakening on co(co)ntraction reductions.

Dereliction on box

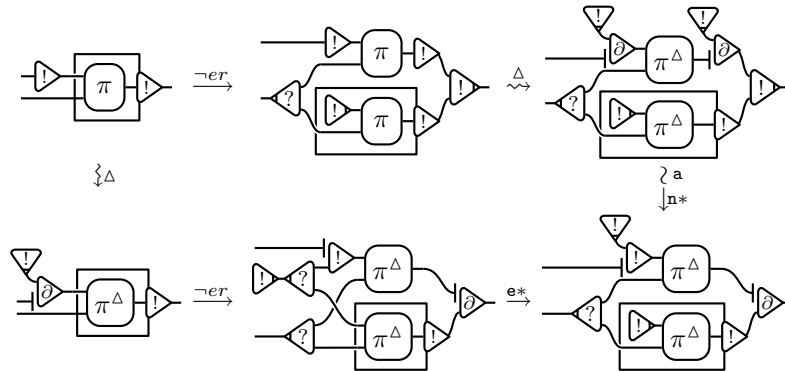


Apply Lemma 7.25 by taking



as π and σ . Notice that $\omega^\Delta[\sigma] \in \text{WN}_{\neg er}$ implies $\omega^\Delta[\mu^\Delta] \in \text{WN}_{\neg er}$ by a trivial simulation of the normalizing chain.

Codereliction on box



By inductive hypothesis, $\omega^\Delta[\mu^\Delta]$ (top right) is weakly normalizable for non erasing steps, which by Proposition 7.15 is then $\text{SN}\sim$. This allows to reduce

a-convert and n-reduce μ^Δ to the bottom right net (let us call it μ') to get rid of the linear substitution and coweakening, still having an $\text{SN}\sim$ (and therefore $\text{WN}_{\neg er}$) polynet. On the other hand, $\omega^\Delta[\lambda^\Delta] \xrightarrow{\neg er, *} \omega^\Delta[\mu']$ where the erasing steps are only “safe” coweakening, where a final application of Lemma 7.22 gives the wanted result. \square

Proof of Theorem 7.2. The proof is now immediate. Take a DiLL proof net π . Then

$$\pi \in \text{WN}_{\neg er} \implies \pi^\Delta \in \text{WN}_{\neg er} \implies \pi^\Delta \in \text{SN}\sim \implies \pi \in \text{SN}\sim,$$

where

- the first implication is the above Lemma 7.26;
- the second is Proposition 7.15, conservation for $\text{DiLL}_{\partial \varrho}$;
- the third is due to $\text{DiLL}_{\partial \varrho}$ simulating DiLL, Lemma 7.10. \square

Chapter 8

Full Resource Calculus

In this section we will redefine Boudol’s λ -calculus with resources [Bou93] extending it with sums and two kinds of non lazy reduction. We will thus have both depletable and perpetual resources, the latter being the main difference from the resource calculus as described in [ER06b] and presented in Section 4.3, so we will here call it the **full resource calculus**. Once again we will totally turn our attention to only natural coefficients, skipping the issues with generic coefficients. Our main novelty is a presentation that “hard codes” the exponential isomorphism of DiLL, so that we can have a calculus without sums inside the arguments.

8.1 The Language

Let \mathcal{P} be the countable set of ports, which we recycle as variables. The set Δ of simple terms of full resource calculus is defined by the grammar

$$\Delta := \mathcal{P} \mid \lambda \mathcal{P} . \Delta \mid \langle \Delta \rangle (\Delta^!)$$

exactly as the resource calculus we presented at page 82. However we define the set of bags as $\Delta^! := \mathcal{M}_{\text{fin}}(\mathbb{A})$ with **arguments** generated by

$$\mathbb{A}_k ::= \Delta \mid \Delta^\infty.$$

A **differential term**, or simply term, is an element of $\mathbb{N}\langle \Delta \rangle$. We will also deal with $\mathbb{N}\langle \Delta^! \rangle$, called **differential bags**. An argument of the form t^∞ is called **perpetual** or **exponential**, otherwise it is **linear**. Again, bags are multisets presented in multiplicative notation. As we did for resource calculus, we will use the notation $\Delta^{(!)}$ (resp. $\mathbb{N}\langle \Delta^{(!)} \rangle$) to mean either simple terms or bags (resp. terms or differential bags).

The above constructors are all extended to sums, all by multilinearity except the one for boxed argument, which is extended by mapping sum to product. More in detail, we add to the linearity equations 4.1 given to resource calculus the following ones

$$(u + v)^\infty = u^\infty v^\infty, \quad 0^\infty = 1.$$

Given a bag A , its **linear part** $\mathcal{L}(A)$ (resp. **perpetual** or **exponential part** $\mathcal{E}(A)$) is the multiset of its linear (resp. exponential) arguments. As usual

terms are considered identical up to α -conversion. A (monic) **context** is a differential term or bag that uses a distinguished variable called its **hole** exactly once, similarly to what was done for nets. Formally the simple contexts $\Delta_{[\]}$ are generated by the grammar

$$\Delta_{[\]} ::= [\] \mid \lambda \mathcal{V} \Delta_{[\]} \mid \langle \Delta_{[\]} \rangle \Delta^! \mid \langle \Delta \rangle \Delta_{[\]}^!$$

with

$$\Delta_{[\]}^! := \Delta_{[\]} \Delta^! \mid (\Delta_{[\]})^\infty \Delta^!.$$

The latter are called **bag contexts**. Finally, a context will be a simple one in $\Delta_{[\]}$ summed to any differential term. Given a context $C[\]$ and a differential term t , $C[t]$ is defined as usual by blindly substituting t for the hole (allowing variable capture), applying the generalizations for sums described above. Clearly for each simple subterm $u \subseteq s$ there exists a unique context $C_{s,u}[\]$ such that $s = C_{s,u}[u]$. A relation R from Δ to $\mathbb{N}\langle\Delta\rangle$ can then be extended to one from $\mathbb{N}\Delta^{(!)}$ to $\mathbb{N}\Delta^{(!)}$ by context closure by setting

$$s \tilde{R} t \iff \exists u \subseteq s, \exists v \in \mathbb{N}\langle\Delta\rangle \mid u R v, t = C_{s,u}[v].$$

The resource calculus presented in Section 4.3 is clearly embedded in full resource calculus: it corresponds to the subset of terms that have no exponential arguments. Classical terms of λ -calculus can instead be embedded in this one by

$$x^* := x, \quad (\lambda x.s)^* := \lambda x.s^*, \quad ((s)t)^* := \langle s^* \rangle (t^*)^\infty.$$

8.2 Giant Step and Baby Step Reductions

In order to define an operational semantics we have to define the substitution operator, which as in differential λ -calculus takes different forms.

Substitution $s[x := t]$ with $s, t \in \mathbb{N}\langle\Delta\rangle$ is defined as usual, applying the generalizations of constructors by multilinearity, so that for example

$$\langle x \rangle x^\infty [x := u + v] = \langle u + v \rangle (u + v)^\infty = \langle u \rangle u^\infty v^\infty + \langle v \rangle u^\infty v^\infty.$$

Linear substitution $\frac{\partial}{\partial x}$ generalizes the one given in Section 4.3. Inductive rules are:

$$\begin{aligned} \frac{\partial y}{\partial x} \cdot t &:= \delta_{x,y} \cdot t, & \frac{\partial \lambda y.u}{\partial x} \cdot t &:= \lambda y. \frac{\partial u}{\partial x} \cdot t \quad \text{with } y \notin t, \\ \frac{\partial \langle r \rangle A}{\partial x} \cdot t &:= \left\langle \frac{\partial r}{\partial x} \cdot t \right\rangle A + \langle r \rangle \frac{\partial A}{\partial x} \cdot t, \\ \frac{\partial A}{\partial x} \cdot t &:= \sum_{a \in A} \left(\frac{\partial a}{\partial x} \cdot t \right) A/a, & \frac{\partial u^\infty}{\partial x} \cdot t &:= \left(\frac{\partial u}{\partial x} \cdot t \right) u^\infty. \end{aligned}$$

The definition for applications and bags can be compacted into

$$\frac{\partial \langle r \rangle A}{\partial x} \cdot t = \left\langle \frac{\partial r}{\partial x} \cdot t \right\rangle A + \sum_{u \in \mathcal{L}(A)} \langle r \rangle \left(\frac{\partial u}{\partial x} \cdot t \right) A/u + \sum_{v^\infty \in \mathcal{E}(A)} \langle r \rangle \left(\frac{\partial v}{\partial x} \cdot t \right) A. \quad (8.1)$$

Note how the linear substitution operator distributes among linear terms, and extracts a linear copy from a boxed argument if needed. This reflects the

derivation property of the exponential in calculus. Given $y = y(x)$ we have $\frac{\partial e^y}{\partial x} = \frac{\partial y}{\partial x} \cdot e^y$. Such substitution is linear in both u and t , and if $x \notin u$ then $\frac{\partial u}{\partial x} \cdot t = 0$.

Non linear and linear substitutions enjoy the same properties found in [ER03], though due to the simpler syntax proofs are somewhat easier. We state the results needed in the definition of the reductions.

Lemma 8.1 (Schwartz). *For $t \in \mathbb{N}\langle \Delta^{(1)} \rangle$, $u, v \in \mathbb{N}\langle \Delta \rangle$, and x, y such that $y \notin u$, then*

$$\frac{\partial}{\partial x} \left(\frac{\partial t}{\partial y} \cdot v \right) \cdot u = \frac{\partial}{\partial y} \left(\frac{\partial t}{\partial x} \cdot u \right) \cdot v + \frac{\partial t}{\partial y} \cdot \left(\frac{\partial v}{\partial x} \cdot u \right).$$

In particular if also $x \notin v$, the second addend is equal to 0 and we have the classic Schwartz's lemma about commutation of partial derivatives.

Proof. Standard induction on t . The case for application will work because of the way we linearize on the fly exponential arguments. Abstraction is trivial. For variable, we have

$$\frac{\partial}{\partial y} \left(\frac{\partial z}{\partial x} \cdot u \right) \cdot v + \frac{\partial z}{\partial y} \cdot \left(\frac{\partial v}{\partial x} \cdot u \right) = \delta_{x,z} \frac{\partial u}{\partial y} \cdot v + \delta_{y,z} \frac{\partial v}{\partial x} \cdot u = \delta_{y,z} \frac{\partial v}{\partial x} \cdot u = \frac{\partial}{\partial x} \left(\frac{\partial z}{\partial y} \cdot v \right),$$

where we have used $x \notin v$. For application let us deal separately with the three addends appearing in (8.1) for $\frac{\partial \langle r \rangle A}{\partial y} \cdot v$.

For the first one we have

$$\begin{aligned} \frac{\partial}{\partial x} \left(\left\langle \frac{\partial r}{\partial y} \cdot v \right\rangle A \right) \cdot u = \\ \left\langle \frac{\partial}{\partial x} \left(\frac{\partial r}{\partial y} \cdot v \right) \cdot u \right\rangle A + \end{aligned} \quad (S_1)$$

$$\sum_{p \in \mathcal{L}(A)} \left\langle \frac{\partial r}{\partial y} \cdot v \right\rangle \left(\frac{\partial q}{\partial x} \cdot u \right) A/q + \quad (S_2)$$

$$\sum_{p^\infty \in \mathcal{E}(A)} \left\langle \frac{\partial r}{\partial y} \cdot v \right\rangle \left(\frac{\partial q}{\partial x} \cdot u \right) A \quad (S_3)$$

and let us call S_1, S_2, S_3 these addends. By induction hypothesis

$$S_1 = \left\langle \frac{\partial}{\partial y} \left(\frac{\partial r}{\partial x} \cdot u \right) \cdot v \right\rangle A + \left\langle \frac{\partial r}{\partial y} \cdot \left(\frac{\partial v}{\partial x} \cdot u \right) \right\rangle A, \quad (S_{11} + S_{22})$$

so let us call these addends S_{11} and S_{12} .

The second one gives

$$\begin{aligned} \frac{\partial}{\partial x} \left(\sum_{p \in \mathcal{L}(A)} \langle r \rangle \left(\frac{\partial p}{\partial y} \cdot v \right) A/p \right) \cdot u = \\ \sum_{p \in \mathcal{L}(A)} \left\langle \frac{\partial r}{\partial x} \cdot u \right\rangle \left(\frac{\partial p}{\partial y} \cdot v \right) A/p + \end{aligned} \quad (L_1)$$

$$\sum_{p \in \mathcal{L}(A)} \langle r \rangle \left(\frac{\partial}{\partial x} \left(\frac{\partial p}{\partial y} \cdot v \right) \cdot u \right) A/p + \quad (L_2)$$

$$\sum_{p \in \mathcal{L}(A)} \sum_{q \in \mathcal{L}(A)/p} \langle r \rangle \left(\frac{\partial p}{\partial y} \cdot v \right) \left(\frac{\partial q}{\partial x} \cdot u \right) A/(pq) + \quad (L_3)$$

$$\sum_{p \in \mathcal{L}(A)} \sum_{q^\infty \in \mathcal{E}(A)} \langle r \rangle \left(\frac{\partial p}{\partial y} \cdot v \right) \left(\frac{\partial q}{\partial x} \cdot u \right) A/p. \quad (L_4)$$

Again by induction hypothesis

$$L_2 = \sum_{p \in \mathcal{L}(A)} \langle r \rangle \left(\frac{\partial}{\partial y} \left(\frac{\partial p}{\partial x} \cdot u \right) \cdot v \right) A/p + \sum_{p \in \mathcal{L}(A)} \langle r \rangle \left(\frac{\partial p}{\partial y} \cdot \left(\frac{\partial v}{\partial x} \cdot u \right) \right) A/p =: L_{21} + L_{22}$$

Note also that L_3 is symmetric in p and q .

Finally the third is

$$\begin{aligned} \frac{\partial}{\partial x} \left(\sum_{p^\infty \in \mathcal{E}(A)} \langle r \rangle \left(\frac{\partial p}{\partial y} \cdot v \right) A \right) \cdot u = \\ \sum_{p^\infty \in \mathcal{E}(A)} \langle r \rangle \left(\frac{\partial r}{\partial x} \cdot u \right) \left(\frac{\partial p}{\partial y} \cdot v \right) A + \end{aligned} \quad (E_1)$$

$$\sum_{p^\infty \in \mathcal{E}(A)} \langle r \rangle \left(\frac{\partial}{\partial x} \left(\frac{\partial p}{\partial y} \cdot v \right) \cdot u \right) A + \quad (E_2)$$

$$\sum_{p^\infty \in \mathcal{E}(A)} \sum_{q \in \mathcal{L}(A)} \langle r \rangle \left(\frac{\partial p}{\partial y} \cdot v \right) \left(\frac{\partial q}{\partial x} \cdot u \right) A/q + \quad (E_3)$$

$$\sum_{p^\infty \in \mathcal{E}(A)} \sum_{q^\infty \in \mathcal{E}(A)} \langle r \rangle \left(\frac{\partial p}{\partial y} \cdot v \right) \left(\frac{\partial q}{\partial x} \cdot u \right) A. \quad (E_4)$$

$$E_2 = \sum_{p^\infty \in \mathcal{E}(A)} \langle r \rangle \left(\frac{\partial}{\partial y} \left(\frac{\partial p}{\partial x} \cdot u \right) \cdot v \right) A + \sum_{p^\infty \in \mathcal{E}(A)} \langle r \rangle \left(\frac{\partial p}{\partial y} \cdot \left(\frac{\partial v}{\partial x} \cdot u \right) \right) A =: E_{21} + E_{22}.$$

Now if we write $\frac{\partial \langle r \rangle A}{\partial x} \cdot u$ as

$$R_1 + R_2 + R_3 := \langle r \rangle \left(\frac{\partial r}{\partial x} \cdot u \right) A + \sum_{q \in \mathcal{L}(A)} \langle r \rangle \left(\frac{\partial q}{\partial x} \cdot u \right) A/q + \sum_{q^\infty \in \mathcal{E}(A)} \langle r \rangle \left(\frac{\partial q}{\partial x} \cdot u \right) A,$$

it becomes straightforward to check that

$$\begin{aligned} \frac{\partial R_1}{\partial y} \cdot v &= S_{11} + L_1 + E_1, \\ \frac{\partial R_2}{\partial y} \cdot v &= S_2 + L_{21} + L_3 + E_3, \\ \frac{\partial R_3}{\partial y} \cdot v &= S_3 + E_{21} + L_4 + E_4, \\ \frac{\partial \langle r \rangle A}{\partial y} \cdot \left(\frac{\partial v}{\partial x} \cdot u \right) &= S_{12} + L_{22} + E_{22}, \end{aligned}$$

which covers the result. \square

If u_1, \dots, u_n are such that $x \notin u_i$, then by the above lemma for any permutation $\sigma \in S_n$ we have

$$\frac{\partial}{\partial x} \left(\dots \left(\frac{\partial}{\partial x} \left(\frac{\partial t}{\partial x} \cdot u_1 \right) \cdot u_2 \right) \dots \right) \cdot u_n = \frac{\partial}{\partial x} \left(\dots \left(\frac{\partial}{\partial x} \left(\frac{\partial t}{\partial x} \cdot u_{\sigma(1)} \right) \cdot u_{\sigma(2)} \right) \dots \right) \cdot u_{\sigma(n)}.$$

This justifies an iterated notation with multisets: suppose A is a linear bag, that is a bag with $\mathcal{E}(A) = 1$, and that $x \notin A$, that is $x \notin u$ for every $u \in |A|$. Then, if $A = u_1 \cdots u_n$, we write

$$\frac{\partial^n t}{\partial x^n} \cdot A := \frac{\partial}{\partial x} \left(\dots \left(\frac{\partial t}{\partial x} \cdot u_1 \right) \dots \right) \cdot u_n$$

which by Schwartz's lemma is well defined regardless of the order in which we write A .

In order to give a uniform presentation to define reduction, we employ one more substitution directly based on the regular one: the **partial substitution** of u for x in t is simply $t[x := x + u]$. When using it we will always imply that $x \notin u$. Then this easily gives results similar to the Schwartz lemma when mixing the two types of substitution.

Lemma 8.2. *If $y \notin u$, then*

$$\frac{\partial}{\partial x} (t[y := y + v]) \cdot u = \left(\frac{\partial t}{\partial x} \cdot u \right) [y := y + v] + \frac{\partial t}{\partial y} \cdot \left(\frac{\partial v}{\partial x} \cdot u \right) [y := y + v].$$

In particular if also $x \notin v$ then $\frac{\partial}{\partial x} (t[y := y + v]) \cdot u = \left(\frac{\partial t}{\partial x} \cdot u \right) [y := y + v]$.

Proof. For $t = z$ variable, note $z[y := y + v] = z + \delta_{y,z} \cdot v$. Then

$$\frac{\partial}{\partial x} (z + \delta_{y,z} \cdot v) \cdot u = \frac{\partial z}{\partial x} \cdot u + \delta_{y,z} \cdot \frac{\partial v}{\partial x} \cdot u = \frac{\partial z}{\partial x} \cdot u + \frac{\partial z}{\partial y} \cdot \left(\frac{\partial v}{\partial x} \cdot u \right) [y := y + v],$$

where the last substitution can be safely applied as y does not appear freely in the term. Abstraction and application are straightforward. \square

In calculus this commutation corresponds to a precise property. Take a regular function $f(x)$. The derivative of f in a is $f'(a)$. If we define $f_a(x) := f(x + a)$, then we easily get the equality (provided a does not depend on x) $f'_a(0) = f'(a)$. What calculus hides is that this equality correspond to a different scheduling of operations – $f'(a)$ is “calculate the derivative and then evaluate in a ”, while $f'_a(0)$ is “evaluate in $x + a$, and then calculate the derivative, and evaluate in 0”. Here it will be possible to distinguish such a scheduling through the baby-step reduction defined below.

Finally, in order to unify the notation, let the **generalized substitution** of a for x in t , with $a = u$ or $a = u^\infty$ an argument, be

$$S_x t \cdot u := \frac{\partial t}{\partial x} \cdot u, \quad S_x t \cdot u^\infty := t[x := x + u].$$

Using partial substitution instead of the regular one allows us to state the following generalized Schwartz’s lemma.

Lemma 8.3. *For $t \in R\langle\Delta\rangle$, a, b arguments and x, y such that $y \notin a$ and $x \notin b$, we have $S_x(S_y t \cdot b) \cdot a = S_y(S_x t \cdot a) \cdot b$.*

Proof. There are three combinations to check (partial-partial, linear-linear and linear-partial). For the first one let $a = u^\infty, b = v^\infty$. When $x \neq y$ it is a known property of substitution. Here however we can also have $x = y$:

$$t[x := x + v][x := x + u] = t[x := x + u + v] = t[x := x + u][x := x + v].$$

The second is Lemma 8.1. The third is Lemma 8.2. \square

Given any bag $A = a_1 \cdots a_{\#A}$ and a variable $x \notin A$, we can define

$$S_x^{\#A} t \cdot A := S_x(\cdots(S_x t \cdot a_1)\cdots) \cdot a_{\#A} = \left(\frac{\partial^{\#L(A)} t}{\partial x^{\#L(A)}} \cdot \mathcal{L}(A) \right) \left[x := x + \sum_{u^\infty \in \mathcal{E}(A)} u \right].$$

We are now ready to define the reductions, which as for resource calculus come in baby-step and giant-step forms. Baby-step is more local and natural, and more close to the reduction defined for Boudol’s calculus. However giant-step, which empties a bag altogether, is the reduction whose bisimulation result reflects the one for the t° translation of λ -calculus and proof nets.

Definition 8.4 (**g** and **b**). **Giant-step β -reduction** (**g**) is generated by context closure of

$$\langle \lambda x.s \rangle A \xrightarrow{\mathbf{g}} S_x^{\#A} s \cdot A[x := 0].$$

Baby-step β -reduction (**b**) is generated by context closure of

$$\langle \lambda x.s \rangle aA \xrightarrow{\mathbf{b}} \langle \lambda x.S_x s \cdot a \rangle A, \quad \langle \lambda x.s \rangle 1 \xrightarrow{\mathbf{b}} s[x := 0].$$

Clearly while there are as many ways as $\#A$ to **b**-reduce a redex, there is only one way to [**b**]-reduce it, namely giving

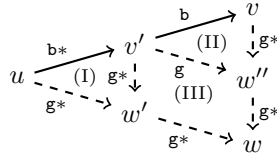
$$S_x^{\#A} s \cdot A[x := 0] = \left(\frac{\partial^{\# \mathcal{L}(A)} s}{\partial x^{\# \mathcal{L}(A)}} \cdot \mathcal{L}(A) \right) \left[x := \sum_{u^\infty \in \mathcal{E}(A)} u \right].$$

Notice that reducing a redex may have duplicating effects at any depth of the term, even greater than the redex itself. However all the properties shown in the previous chapters let us easily deal with such duplications.

Partial substitutions break strong confluence of the sum reduction. However we will be able to infer confluence for **g** (Corollary 8.17) using the translation in nets. We here derive from it the confluence of **b**.

Lemma 8.5. *We have $\mathbf{b}^* \subseteq \mathbf{g}^*$.*

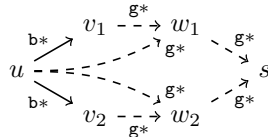
Proof. Let $u \xrightarrow{\mathbf{b}^*} v$. We proceed by induction on the length of such reduction. If it is zero, the result is trivial. Otherwise:



We have (I) by inductive hypothesis, (II) is clear from the definition, as **g**-reducing a redex before or after a single step of **b** on the same redex is the same. The second **g**-reduction appears multiple times as we may have to **g**-reduce in all copies possibly arisen by additive substitution. Then (III) is confluence of **g**. \square

Theorem 8.6. *The baby-step β -reduction is confluent.*

Proof. Suppose $u \xrightarrow{\mathbf{b}^*} v_1, v_2$. We get the following confluence diagram:



The left triangles are from the above lemma, while the right square is simply confluence of **g**. As $[g^*]$ is contained in \mathbf{b}^* , we get the result. \square

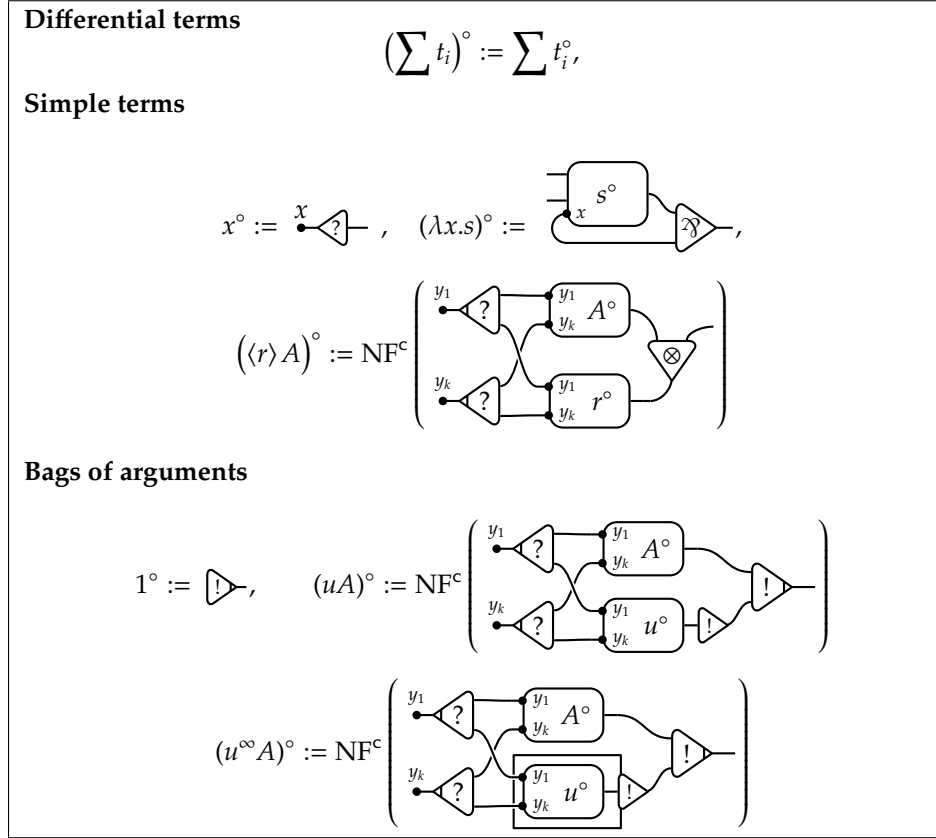


Figure 8.1: Inductive rules for the definition of t° for full resource calculus.

8.3 The Translation

We will now extend both the translation of λ -terms to MELL λ -nets [Dan90, Reg92] presented in Section 4.2.3 and that of resource calculus in DiLL₀ λ -nets as done in [ER06b] and presented in Section 4.3 to one from full resource calculus to DiLL λ -nets.

We will draw all nets with the $o/!o$ conclusion right and the rest left, so types will be omitted. If we label a port with a set of ports, it must be intended to stand for multiple ports, together with the cells above (we will use it only with contractions). And once again, \approx will denote the equivalence modulo weakened conclusions.

8.3.1 Statics: Definition and Sequentialization

Using the rules in Figure 8.1 for each t term (resp. bag or argument) we define t° , a net with conclusions $x_1 : ?i, \dots, x_k : ?i, o$ (resp. $!o$) where the free ports x_1, \dots, x_n contain the free variables in t (in fact the free variable of t are the free ports that are in all \approx -representants of t°). The fact that t° is indeed correct is straightforward. Adding freely weakened conclusions is used in the definition to unify the sets of variables. It is important to note that the translation is well defined with respect to equality modulo weakened conclusions only because

of pull reductions performed on boxes. Also we see that the translation is a function only on \mathbf{a} -equivalence classes, because of the possible ways to order a multiset. Finally, the translation is compatible with the generalizations of the constructors to sums, with the special case of a perpetual argument covered by $\overset{\text{sp}}{\sim}$ for a proper sum and \mathbf{z} reduction for the 0 case.

Remark 8.7. For every term t its translation t° is \mathbf{ec} -normal. Moreover each redex in t corresponds exactly to an \mathbf{m} -redex in t° . So in fact t is normal iff t° is normal.

Also we easily see that if u is a term of plain resource calculus the translation coincides with Ehrhard and Regnier's one, and if t is a term of ordinary λ -calculus then $(t^*)^\circ = t^\circ$, the translation of Danos and Regnier. As in that one, sequentialization (i.e. surjectivity) is valid for nets without exponential axioms.

Theorem 8.8 (sequentialization of \mathbf{ec} -normal nets). *For every \mathbf{ec} -normal DiLL λ -net π with no exponential axiom there is uniquely either a term t or a bag A such that $t^\circ = \pi$ (resp. $A^\circ = \pi$), modulo weakened conclusions and \sim .*

Proof. Let's reason by induction on the depth of π , and suppose it is simple (afterwards we can sum again the terms obtained). We can suppose by $\overset{\text{asp}}{\sim}$ conversion that all box contains a simple net and that no contraction tree has two or more leaves on a same box.

Assign a port name to each \mathfrak{A} at depth 0. As there are no exponential cuts every dereliction and auxiliary port at depth 0 in π has only one $?$ -path coming out of it, and it can end only on a \mathfrak{A} or a conclusion. In the We then label each dereliction with the name of the port or \mathfrak{A} to which this path leads, and similarly for the conclusions at depth 1.

We use the inductive hypothesis on the box contents, naming their free ports with such labels (by the conversion we did above no name is repeated), and get for each box a term translating into its contents.

Now take a principal switching of π . Delete all weakenings at depth 0, that because of \mathbf{a} -normality and the particular switching we have chosen make up each a connected component in itself. Delete also all downward $?$ -paths and the cells internal to it: we delete all contraction trees, as by the above labelling we need not to remember these sharing anymore. By the hypothesis on axioms, the wires above tensors and cocontractions cannot be deleted by this procedure. We are left with an acyclic connected graph, which therefore is a tree. If we take the unique o or $!o$ conclusion as a root, the leaves are necessarily labelled derelictions or boxes.

What we obtained is in fact the syntactic tree of a term, as we replace

- each \mathfrak{A} labelled by x is a λx node,
- each \otimes cell by an application node where the tree on the principal port goes in functional position and the one on the $!o$ port in argument position,
- each cocontraction is a multiset product of the above trees,
- each box with t^∞ where t is what (by inductive hypothesis) translates into its contents,
- each codereliction is erased,

This term is easily seen as translating into the same net we started from. \square

Notice that the property about exponential axioms is not stable under one-step e-reduction, because of the contraction-cocontraction step. One could

- either consider on the contrary only η -contracted nets, i.e. nets without *identity boxes*, boxes containing a single wire with a dereliction; this has also the positive side effect on cutting down on reduction steps;
- or substitute the contraction on cocontraction rule with one having four boxes on the wires, to the detriment of both readability and reduction steps.

However let us say that π is **essentially without exponential axioms** if $NF^e(\pi)$ is without exponential axioms. Then this property is stable by reduction, as we will prove in Lemma 8.15.

8.3.2 Dynamics: Bisimulation

We want to show that reductions in the two systems are strongly linked by this translation. This is done in two steps, showing the two directions of bisimulation. First we have to state a substitution lemma.

Lemma 8.9 (argument substitution). *Given an argument a , a simple term u and a variable $x \notin a$, we have that*

Proof. Clearly we have to deal separately with the two possible types of substitution.

First suppose $x \notin u$, which implies (by canonicity) that x is a weakened conclusion. Then

and

where we have thus covered the cases where $a = t^\infty$ and $a = t$ linear.

Suppose therefore $x \in u$, and let us reason by induction on u .

If u is a variable then $u = x$. Then, if $a = t^\infty$

while for $a = t$ linear

$$\begin{array}{c}
 \begin{array}{c}
 s \\
 x
 \end{array}
 \begin{array}{c}
 \triangleleft \\
 \triangleleft \\
 \circlearrowleft \\
 \triangleleft \\
 \triangleleft
 \end{array}
 \begin{array}{c}
 ? \\
 ? \\
 t^\circ \\
 ? \\
 ?
 \end{array}
 \begin{array}{c}
 \longrightarrow \\
 \longrightarrow \\
 \longrightarrow \\
 \longrightarrow \\
 \longrightarrow
 \end{array}
 \begin{array}{c}
 \triangleleft \\
 \triangleleft \\
 \triangleleft \\
 \triangleleft \\
 \triangleleft
 \end{array}
 \begin{array}{c}
 ? \\
 ? \\
 ? \\
 ? \\
 ?
 \end{array}
 \end{array}
 \xrightarrow{\text{en}}
 \begin{array}{c}
 s \\
 x
 \end{array}
 \begin{array}{c}
 \triangleleft \\
 \circlearrowleft \\
 \triangleleft
 \end{array}
 \begin{array}{c}
 ? \\
 t^\circ \\
 ?
 \end{array}
 = \left(\frac{\partial x}{\partial x} \cdot t \right)^\circ$$

The inductive step for abstraction (as well as for linear argument) is immediate.

If $u = \langle r \rangle A$ then and σ is the starting net then we have

$$\begin{array}{c}
 \sigma \xleftarrow{\text{cs}^*} \begin{array}{c} s \\ x \end{array} \begin{array}{c} \triangleleft \\ \triangleleft \\ \circlearrowleft \\ \triangleleft \\ \triangleleft \end{array} \begin{array}{c} ? \\ ? \\ t^\circ \\ ? \\ ? \end{array} \begin{array}{c} \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \end{array} \begin{array}{c} \triangleleft \\ \triangleleft \\ \triangleleft \\ \triangleleft \\ \triangleleft \end{array} \begin{array}{c} ? \\ ? \\ ? \\ ? \\ ? \end{array} \begin{array}{c} S_x A^\circ \\ S_x r^\circ \end{array} \\
 \xrightarrow{\text{en}^* \text{ a}} \begin{array}{c} s \\ x \end{array} \begin{array}{c} \triangleleft \\ \triangleleft \\ \circlearrowleft \\ \triangleleft \\ \triangleleft \end{array} \begin{array}{c} ? \\ ? \\ t^\circ \\ ? \\ ? \end{array} \begin{array}{c} \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \end{array} \begin{array}{c} \triangleleft \\ \triangleleft \\ \triangleleft \\ \triangleleft \\ \triangleleft \end{array} \begin{array}{c} ? \\ ? \\ ? \\ ? \\ ? \end{array} \begin{array}{c} A^\circ \\ r^\circ \end{array} \\
 \xrightarrow{\text{ec}} \text{NF}^c \left(\begin{array}{c} s \\ x \end{array} \begin{array}{c} \triangleleft \\ \triangleleft \\ \circlearrowleft \\ \triangleleft \\ \triangleleft \end{array} \begin{array}{c} ? \\ ? \\ t^\circ \\ ? \\ ? \end{array} \begin{array}{c} \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \end{array} \begin{array}{c} \triangleleft \\ \triangleleft \\ \triangleleft \\ \triangleleft \\ \triangleleft \end{array} \begin{array}{c} ? \\ ? \\ ? \\ ? \\ ? \end{array} \begin{array}{c} (A[x := t])^\circ \\ (r[x := t])^\circ \end{array} \right) = (\langle r[x := t] \rangle A[x := t])^\circ
 \end{array}$$

which is what we were looking for. The first expansion is due to the definition of the translation, while in the last chain of reductions induction hypothesis is used. Church-Rosser modulo \sim ensures the normal form at the end is the the same of the starting net.

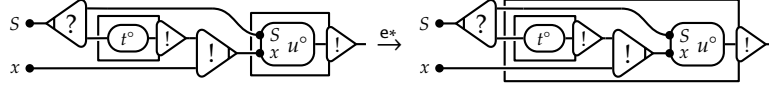
For the same case with linear substitution, we have similarly

$$\begin{array}{c}
 \sigma \xleftarrow{\text{cs}^*} \begin{array}{c} s \\ x \end{array} \begin{array}{c} \triangleleft \\ \triangleleft \\ \circlearrowleft \\ \triangleleft \\ \triangleleft \end{array} \begin{array}{c} ? \\ ? \\ t^\circ \\ ? \\ ? \end{array} \begin{array}{c} \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \end{array} \begin{array}{c} \triangleleft \\ \triangleleft \\ \triangleleft \\ \triangleleft \\ \triangleleft \end{array} \begin{array}{c} ? \\ ? \\ ? \\ ? \\ ? \end{array} \begin{array}{c} S_x A^\circ \\ S_x r^\circ \end{array} \\
 \xrightarrow{\text{en}^* \text{ a}} \begin{array}{c} x \\ s \end{array} \begin{array}{c} \triangleleft \\ \triangleleft \\ \circlearrowleft \\ \triangleleft \\ \triangleleft \end{array} \begin{array}{c} ? \\ ? \\ t^\circ \\ ? \\ ? \end{array} \begin{array}{c} \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \end{array} \begin{array}{c} \triangleleft \\ \triangleleft \\ \triangleleft \\ \triangleleft \\ \triangleleft \end{array} \begin{array}{c} ? \\ ? \\ ? \\ ? \\ ? \end{array} \begin{array}{c} A^\circ \\ r^\circ \end{array} \\
 + \\
 \begin{array}{c} s \\ x \end{array} \begin{array}{c} \triangleleft \\ \triangleleft \\ \circlearrowleft \\ \triangleleft \\ \triangleleft \end{array} \begin{array}{c} ? \\ ? \\ t^\circ \\ ? \\ ? \end{array} \begin{array}{c} \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \end{array} \begin{array}{c} \triangleleft \\ \triangleleft \\ \triangleleft \\ \triangleleft \\ \triangleleft \end{array} \begin{array}{c} ? \\ ? \\ ? \\ ? \\ ? \end{array} \begin{array}{c} A^\circ \\ r^\circ \end{array} \\
 \xrightarrow{\text{ec}} \left(\left(\frac{\partial r}{\partial x} \cdot t \right) A + \langle r \rangle \frac{\partial A}{\partial x} \cdot t \right)^\circ
 \end{array}$$

The same steps apply for multiplication of bags.

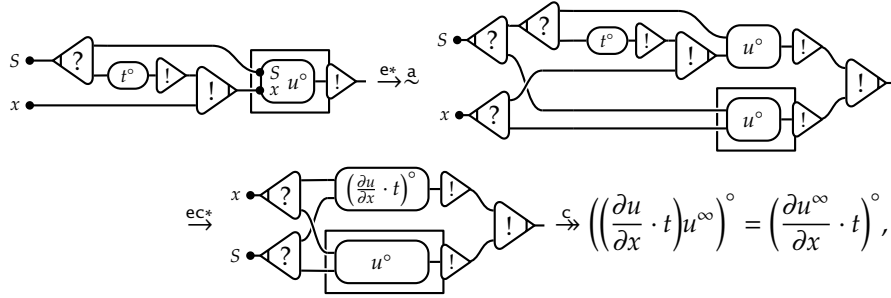
Therefore we are left with the case of the perpetual arguments. Observe first that the translation respects the equalities $(\infty s + t) = s^\infty t^\infty$ and $0^\infty = 1$ by means of the bang sum equivalence and the bang zero reduction, so we can safely apply induction hypothesis to the contents of the box, even if the result is 0 or a sum.

For $a = u^\infty$ we have



which p-converts (taking the contraction inside the box) and ec-normalizes to $(u[x := x + t])^\infty$.

For $a = u$ linear we have



which concludes the proof. \square

Lemma 8.10 (0 substitution). *Given an argument a , a simple term or bag u and a variable $x \notin a$, we have that*

$$s \cdot \left[\begin{array}{c} \bullet \\ \downarrow \\ \text{!} \\ \downarrow \\ x \end{array} \right] u^\circ \xrightarrow{\text{ec}} u[x := 0]$$

Proof. Straightforward by the rules of coweakening. \square

We can now prove the main substitution lemma.

Lemma 8.11 (substitution). *If A is a bag of arguments and u is a simple term, then*

$$s \cdot \left[\begin{array}{c} \bullet \\ \downarrow \\ \text{!} \\ \downarrow \\ s \end{array} \right] A^\circ \cdot \left[\begin{array}{c} \bullet \\ \downarrow \\ \text{!} \\ \downarrow \\ x \end{array} \right] u^\circ \xrightarrow{\text{ec}} (S_x^{\#A} u \cdot A)[x := 0].$$

Proof. Let σ be the net under consideration. If $A = a_1 \cdots a_n$, then we have

$$\sigma \xleftarrow{\text{n}} \left[\begin{array}{c} \bullet \\ \downarrow \\ \text{!} \\ \downarrow \\ s \end{array} \right] a_n^\circ \cdot \left[\begin{array}{c} \bullet \\ \downarrow \\ \text{!} \\ \downarrow \\ s \end{array} \right] a_2^\circ \cdot \left[\begin{array}{c} \bullet \\ \downarrow \\ \text{!} \\ \downarrow \\ s \end{array} \right] a_1^\circ \cdot \left[\begin{array}{c} \bullet \\ \downarrow \\ \text{!} \\ \downarrow \\ x \end{array} \right] u^\circ =: \sigma'$$

where we do not draw all the contraction trees joining together the S ports. The initial n-expansion is not there only if $n = 0$.

Now a repeated application of Lemma 8.9 and a final one of Lemma 8.10 gives

$$\sigma \xleftarrow{\text{n}} \xrightarrow{\text{ec}} (S_x^{\#A} s \cdot A[x := 0])^\circ.$$

By $\text{CR}\sim$, we get that $\text{NF}^{\text{ec}}(\sigma)$ is the member on the left (up to \sim). \square

Note how the reduction on nets involved in the next theorem has a particular shape, so that even if the result is a logical equivalence it does not yet mean that one can compute with nets. A priori there could be reductions in nets which do not correspond to any reduction in terms. Such result is truly achieved by the result after it.

Proposition 8.12 (one step simulation). $s^\circ \xrightarrow{\mathfrak{m}}^{\text{ec}} \sigma$ if and only if $\sigma = t^\circ$ (up to \sim) and $s \xrightarrow{\mathfrak{g}} t$.

Proof. Because of a minor technical point we cannot yet extract t directly from σ with the sequentialization theorem, as we are not sure σ has no exponential axioms.

The if part is a direct consequence of Lemma 8.11, as firing the \mathfrak{m} -redex corresponding to the redex fired in s makes the net fall into the hypothesis of the substitution lemma.

Vice versa let $s \xrightarrow{\mathfrak{g}} t$ be the result of firing the redex corresponding to the multiplicative cut fired at the beginning of the reduction $s \xrightarrow{\mathfrak{m}} \pi \xrightarrow{\text{ec}} \sigma$. Because of the first part of the proof, $s^\circ \xrightarrow{\mathfrak{m}} \pi \xrightarrow{\text{ec}} t^\circ$, where the intermediate step π must be the same as before. By uniqueness of normal form we get $\sigma = t^\circ$. \square

We get a first feedback on the properties of the calculus, as the above result suffices to show strong normalization of the calculus. We will not go into the details of defining the typing of full resource calculus. Let it suffice to say that an intuitionistic typing on a term s lifts to a MELL typing of s° .

Corollary 8.13. *If s° is strongly normalizing in DiLL, then so is t for $\xrightarrow{\mathfrak{g}}$ reduction. In particular simply typed full resource calculus is strongly normalizing.*

Proof. The above proposition states that if $u \xrightarrow{\mathfrak{g}} v$ then $u^\circ \xrightarrow{\text{mec}+} v^\circ$, so no infinite chains are possible from s , as they are simulated by infinite chains from s° .

As DiLL simply typed nets are strongly normalizing modulo \sim , we also get strong normalization of all typed terms. \square

Before going on, let us prove an intermediate lemma that generalizes the above result.

Lemma 8.14. *If $s^\circ \xrightarrow{\mathfrak{m}^*}^{\text{ec}} \sigma$ then $\sigma = t^\circ$ and $s \xrightarrow{\mathfrak{g}^*} t$.*

Proof. Let M be the sequence of multiplicative reductions in the reduction $R : s^\circ \xrightarrow{\mathfrak{m}^*} \pi \xrightarrow{\text{ec}} t^\circ$. Let us reason by structural induction on s .

If s is a differential term we can reason independently on its simple terms. If s is a variable there is no redex and the result is trivial. Also passing to the induction hypothesis if $s = \lambda x.u$ is an abstraction is easy, as all reductions in the net cannot touch the terminal \mathfrak{A} -cell which corresponds to the abstraction.

Take the case of an application $\langle r \rangle a_1 \cdots a_n$. First suppose that all reductions in M happen either in r° or in either of the a_i° s. We can partition M into $L : r^\circ \xrightarrow{\mathfrak{m}^*} \nu$ and $N_i : a_i^\circ \xrightarrow{\mathfrak{m}^*} \tau_i$, and we can freely commute all reductions which happen in different subnets. By *ec*-normalizing ν and the τ_i s we get, using induction hypothesis, $r^\circ \xrightarrow{\mathfrak{L}}^{\text{e}} s^\circ$ and $a_i^\circ \xrightarrow{N_i}^{\text{e}} b_i^\circ$ with $r \xrightarrow{\mathfrak{g}^*} s$ and $a_i \xrightarrow{\mathfrak{g}^*} b_i$.

Applying these reductions on the whole $(\langle r \rangle a_1 \cdots a_n)^\circ$ we can commute L and N_i back into their place in M , possibly using $\xrightarrow{\Sigma \mathfrak{m}}$ versions of them. The

exponential reductions on one subnet cannot duplicate the redexes in the other if not by splitting the whole term with a sum. Then the reduction substitutes each of the subnets by the translation of their redexes:

$$\langle r \rangle a_1 \cdots a_n \circ \xrightarrow{M} \text{ec} \langle s \rangle b_1 \cdots b_n \circ,$$

and $\langle r \rangle a_1 \cdots a_n \xrightarrow{\mathfrak{g}^*} \langle s \rangle b_1 \cdots b_n$.

Suppose now that M reduces also a multiplicative cut outside of r° or either a_i . This means $s = \langle r \rangle A$ is itself a redex, with $r = \lambda x.u$. Reducing the corresponding multiplicative cut cannot create other multiplicative cuts, (the multiplicative wire is on a conclusion) so that we can commute it back and forth along M . We can still partition the multiplicative reduction into $L : u^\circ \xrightarrow{m^*} \rho$, $N_i : a_i \xrightarrow{m^*} \pi_i$ and the single reduction μ on the external cut. If we exclude μ from the reduction we have by a reasoning identical to the one above

$$\langle \lambda x.u \rangle a_1 \cdots a_n \circ \xrightarrow{L} \text{N}_1 \xrightarrow{\dots} \text{N}_n \text{ec} \langle \lambda x.v \rangle b_1 \cdots b_n \circ$$

with $s \xrightarrow{\mathfrak{g}^*} \langle \lambda x.v \rangle B$ (as $u \xrightarrow{\mathfrak{g}^*} v$ and $a_i \xrightarrow{\mathfrak{g}^*} b_i$).

Now in $\langle \lambda x.v \rangle B \circ$ we execute μ and then *ec*-normalize, and by Proposition 8.12 we get $\left(\left(S_x^{\#A} v \cdot B \right) [x := 0] \right)^\circ$. By commuting all multiplicative reductions back into their place in M and before the exponential-canonical ones (the ones triggered by μ remain at the bottom), by uniqueness of *ec*-normal form we conclude that $\left(S_x^{\#B} v \cdot B \right) [x := 0]$ (to which s reduces) translates into t . \square

Lemma 8.15. *If for a net π we have $\text{NF}^{\text{ec}}(\pi) = s^\circ$ (equivalent by sequentialization to π being essentially without exponential axioms) $\pi \xrightarrow{\text{mec}} \sigma$ then $\text{NF}^{\text{ec}}(\sigma) = t^\circ$ with $s \xrightarrow{\mathfrak{g}^*} t$ (in particular also σ is essentially without exponential axioms).*

Proof. If $\pi \xrightarrow{\text{ec}} \sigma$ there is nothing to prove, as the two normal forms coincide. If instead $\pi \xrightarrow{m} \sigma$ we get

$$\begin{array}{ccc} \pi & \xrightarrow{m} & \sigma \\ \text{ec} \downarrow & & \text{ec} \downarrow \\ \text{NF}^{\text{ec}}(\sigma) & \xrightarrow{m^*} & \sigma' \xrightarrow{\text{ec}} \text{NF}^{\text{ec}}(\pi) \end{array}$$

The left square is commutation of m and *ec* (Lemma 6.11), while the right triangle is confluence to the *ec*-normal form. By sequentialization, as $\text{NF}^{\text{ec}}(\pi)$ is without exponential axioms, we get $\text{NF}^{\text{ec}}(\pi) = s^\circ$, which combined with the above result gives $\text{NF}^{\text{ec}}(\sigma) = t^\circ$, which in turn gives that σ is essentially without exponential axioms. \square

Theorem 8.16 (giant-step bisimulation). *If $s^\circ \xrightarrow{\text{mec}^*} \text{ec} \sigma$ if and only if $\sigma = t^\circ$ and $s \xrightarrow{\mathfrak{g}^*} t$.*

Proof. The if part is just iteration of Proposition 8.12. For the only if part let

$$s^\circ = \pi_0 \xrightarrow{\text{mec}} \pi_1 \xrightarrow{\text{mec}} \dots \xrightarrow{\text{mec}} \pi_n = \sigma$$

be the reduction taken into account, let $\sigma_i := \text{NF}^{\text{ec}}(\pi_i)$, with therefore $\sigma_0 = \pi_0$ and $\sigma_n = \sigma$.

By repeatedly applying the previous lemma we get $\sigma_i = s_i^\circ$ with $s_i \xrightarrow{\mathbf{g}^*} s_{i+1}$ which composed give what we are looking for. \square

We can now get confluence of full resource calculus deriving it from what was proved for DiLL.

Corollary 8.17. *The reduction \mathbf{g} on terms is confluent.*

Proof. Take $s \xrightarrow{\mathbf{g}^*} u, v$. By simulation $s^\circ \xrightarrow{\text{mec}^*} u^\circ, v^\circ$, so that by confluence of the mec reduction we further get $u^\circ, v^\circ \xrightarrow{\text{mec}^*} \pi$. We then have $u^\circ, v^\circ \xrightarrow{\text{mec}^*} \text{NF}^{\text{e}}(\pi)$ which by full simulation gives $\text{NF}^{\text{e}}(\pi) = t^\circ$ and $u, v \text{redto}[g^*]t$. \square

Annexes

Index

- ∂ , 136
- \preceq , 116
- ϱ , 136
- MALL net, 48

- additive, 50
- additive contraction, 95
- α -conversion, 19
- α -equivalence
 - on nets, 19
- anti-input, 76
- anti-output, 76
- argument, 154
 - bags
 - exponential part, 154
 - linear part, 154
 - differential bags, 154
 - exponential, 154
- arity, 17
- axiom, 19

- bag of arguments, 82
- box, 30

- cell, 17
- cell structural equivalence, 35
- coarity, 17
- codepth, 31
- coherence, 60
- coherent space, 59
- compatible paths, 96
- context, 31
 - linear, 24
 - polycontext, 28
 - term, 155
- correction, 46
 - strong, 45
- correctness
 - graph, 52
 - semantic, 41
- correctness graph, 95

- critical pair, 34
- cut, 19
- cut formula, 57
- cut sequent, 57

- dead end, 146
- deadlocks, 17
- degree, 17
- depth, 31

- equivalence
 - associative, 75
 - bang sum, 86
 - push, 75
- experiment, 39
- exponential, 72
- exposure, 105

- flat nets, 137

- glueing, 24

- hypercliques, 64
- hypercoherence, 64
- hypercoherent space, 64
- hypercorrectness, 96
 - weak, 97

- input, 76
- interface, 24

- labelling, 137
- λ -net, 77
- λ -calculus, 78
- linking, 47
- linkings, 23

- module, 24

- net, 17
 - η -expanded, 26, 27
 - slice net, 29

- output, 76
- path
 - exponential, 112
- polycontext, 31
- polymodule, 28
- polynet, 27
- port, 17
 - active and passive, 17
 - auxiliary, 17
 - bounded, 17
 - free, 17
 - principal, 17
- proof, 44
- proof nets
 - MALL, 48
- pure, 74
- redex, 33
 - orthogonal, 34
- reduction
 - baby step, 159
 - baby-step, 83
 - bang zero, 86
 - canonical, 75
 - erasing and non erasing, 133
 - exponential, 72
 - giant step, 159
 - giant-step, 84
 - neutral, 75
 - non erasing, 74
 - pull, 75
 - sum, 37
- reduction system, 33
- residual, 28
- resolution, 49
 - additive, 49
 - proper, 50
 - $\&$ -, 50
- saturated
 - MALL net, 55
 - set of MALLnets, 54
- sequent, 26, 44
- shared cut reduction, 58
- signature, 17
- spread, 113
- state, 41
- strictly $\&$ -oriented, 96
- subject reduction, 35
- substitution
 - generalized, 158
 - linear, 83, 155
 - partial, 157
- switching, 45
 - acyclicity, 46
 - graph, 45
 - path, 46
- term
 - differential, 82, 154
- toggling, 53
- tree, 23
- type
 - atomic, 27
- type system, 25
 - axiomatic, 26
 - typing, 25
- webbed, 41
- weight
 - ?-weight, !-weight, 113
- wire, 17
 - dormant, 18
 - external, 17
 - internal, 17

List of Figures

2.1	An example of net.	18
3.1	Cells, typings and reductions of MLL.	43
3.2	Inference and desequentialization rules for proofs of MLL.	44
3.3	The mix inference and desequentialization rules.	46
3.4	Cells, typing rules and reductions of MALL nets. We also show the alternative and intuitive notation, where the symbols $\&_1$ and $\&_2$ (resp. \oplus_1 and \oplus_2) are identified, but recovered from the position of the auxiliary port: left for $\&_1$ (resp. \oplus_1) and right for $\&_2$ (resp. \oplus_2).	48
3.5	Sequent calculus rules for MALL additive connectives, and the respective desequentialization rules for nets.	49
3.6	An example of MALL net, showing the compact representation.	51
3.7	Sequent calculus rules for MALL_{cut}	57
3.8	Rules for the experiments for all the cells of MALL.	59
3.9	The Gustave MALL net γ is shown in Figure (b). P stands for $\alpha^\perp \otimes \beta^\perp$, and the link with three wires is shorthand for the trivial linking shown in Figure (a).	63
3.10	The proof structure δ : an unsequentializable structure such that $\llbracket \delta \rrbracket$ is a hyperclique.	67
4.1	Cells of MELL, together with typing rules for cells and for the box and the reduction rules. Recall that in the box on box reduction the two boxes must be different (a condition that is not trivial in incorrect MELL nets).	73
4.2	Examples of non-termination (a), and non-confluence (b). Multiplicative units are not really necessary, but simplify the examples.	73
4.3	The generating pairs for associative and push equivalences, and the neutral and pull reductions.	75
4.4	Examples of infinite reductions modulo \approx and \approx . The use of the multiplicative unit 1 simplifies the counterexamples but is not required.	76
4.5	Typing rules for pure λMELL	77
4.6	Inductive rules for the definition of t° . Labels give names to ports, which are identified with the variables of the translated term. Conversion modulo \approx is implicitly used.	79
4.7	The additional cells, typing and reduction rules for differential interaction nets, DiLL_0 . Rules for $!$ and $?$ are totally symmetric.	81

4.8	Inductive rules for the definition of t° for resource calculus.	85
4.9	Additional exponential reduction rules for DiLL.	86
4.10	The pairs generating the push and sum equivalences, and the pull and zero reductions.	87
4.11	Example of looping e-reduction with the reversal of the z-reduction.	88
4.12	A counter example to being weakly normalizable by non erasing steps being equivalent to being strongly normalizing.	89
4.13	Rules for lax typing.	90
5.1	Two examples of our version of correctness graphs. The first one shows the rejection of the Gustave net by the criterion, while the second structure, is hypercorrect.	98
6.1	The marking rules of DiLL ^o	111
6.2	Confluence diagram of codereliction on box on dereliction.	129
6.3	Confluence diagram of codereliction on box on contraction. The + . . . part indicates a symmetric addend.	129
6.4	Reduction of a box with two coderelictions on it. Starting with codereliction b swaps the two linear copies of the box contents and therefore both the cocontraction and contraction trees in the last addend.	130
6.5	Coherence diagram between the bang sum equivalence and a codereliction.	132
7.1	The cells ∂ and ϱ of DiLL _{$\partial\varrho$} , and the translation from and to DiLL. The cells are <i>not</i> commutative, but may be drawn with swapped inactive ports, hence the special marking of the left port. ϱ is switching, ∂ is not.	136
7.2	Reduction rules of labelled DiLL _{$\partial\varrho$} ^o . All combinations not shown are the same as in DiLL, with labels left unchanged.	138
7.3	The new associative equivalences.	138
7.4	Reduction of the critical peak with two ∂ cells on a box. The box has only one other auxiliary port for brevity. Starting with the other ∂ cell is completely symmetric, and an a-conversion equates the two results. Labels are omitted as they are unaffected.	140
7.5	Confluence diagram of the ∂ on box on ϱ critical peak.	141
7.6	Coherence diagram of a ∂ cell on an s-conversion.	142
8.1	Inductive rules for the definition of t° for full resource calculus.	160

Bibliography

- [AM99] Samson Abramsky and Paul-André Melliès. Concurrent games and full completeness. In *LICS*, pages 431–442. IEEE Computer Society Press, 1999.
- [Bar84] Henk Barendregt. *The lambda calculus, its syntax and semantics*. Number 103 in Studies in Logic and the Foundations of Mathematics. North-Holland, second edition, 1984.
- [BCL99] Gérard Boudol, Pierre-Louis Curien, and Carolina Lavatelli. A semantics for lambda calculi with resources. *MSCS*, 9(4):437–482, 1999.
- [BE91] Antonio Bucciarelli and Thomas Ehrhard. Sequentiality and strong stability. In *LICS*. IEEE Computer Society Press, 1991.
- [BE01] Antonio Bucciarelli and Thomas Ehrhard. On phase semantics and denotational semantics: the exponentials. *Ann. Pure Appl. Logic*, 109(3):205–241, 2001.
- [Béc98] Denis Béchet. Minimality of the correctness criterion for multiplicative proof nets. *Mathematical Structures in Computer Science*, 8(6):543–558, 1998.
- [Ber78] Gérard Berry. Stable models of typed lambda-calculi. In *International Colloquium on Automata, Languages and Programming*, volume 62 of *Lecture Notes in Computer Science*. Springer, 1978.
- [BHS05] Richard Blute, Masahiro Hamano, and Philip J. Scott. Softness of hypercoherences and MALL full completeness. *Ann. Pure Appl. Logic*, 131(1-3):1–63, 2005.
- [Bie94] Gavin Bierman. *On intuitionistic linear logic*. PhD thesis, University of Cambridge Computer Laboratory, December 1994.
- [Bla92] Andreas Blass. A game semantics for linear logic. *Annals of Pure and Applied Logic*, 56:183–220, April 1992.
- [Bou93] Gérard Boudol. The lambda-calculus with multiplicities. INRIA Research Report 2025, 1993.
- [CKP03] Roberto Di Cosmo, Delia Kesner, and Emmanuel Polonovski. Proof nets and explicit substitutions. *Mathematical Structures in Computer Science*, 13(3):409–450, 2003.

- [Dan90] Vincent Danos. *La Logique Linéaire appliquée à l'étude de divers processus de normalisation (principalement du λ -calcul)*. Thèse de doctorat, Université Paris VII, 1990.
- [dC07] Daniel de Carvalho. *Sémantiques de la logique linéaire et temps de calcul*. PhD thesis, Université Aix-Marseille II, 2007. Thèse de Doctorat.
- [DCG99] Roberto Di Cosmo and Stefano Guerrini. Strong normalization of proof nets modulo structural congruences. *LNCS*, 1631:75–??, 1999.
- [DCK97] Roberto Di Cosmo and Delia Kesner. Strong normalization of explicit substitutions via cut elimination in proof nets. In *LICS*, page 35. IEEE Computer Society, 1997.
- [DdW94] Eric Duquesne and Jacques Van de Wiele. Modèles cohérents des réseaux purs. *Archive for Mathematical Logic*, 33(2):131–158, 1994.
- [DG08] Paolo Di Giamberardino. *Jump from parallel to sequential proofs: on polarities and sequentiality in Linear Logic*. PhD thesis, Università Roma Tre / Université Aix-Marseille II, 2008.
- [DR89] Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.
- [Ehr95] Thomas Ehrhard. Hypercoherence: A strongly stable model of linear logic. In *Advances in Linear Logic*, pages 83–108. Cambridge University Press, 1995.
- [Ehr02] Thomas Ehrhard. On Köthe sequence spaces and linear logic. *MSCS*, 12:579–623, 2002.
- [Ehr05] Thomas Ehrhard. Finiteness spaces. *Mathematical Structures in Comp. Sci.*, 15(4):615–646, 2005.
- [EL07] Thomas Ehrhard and Olivier Laurent. Interpreting a finitary pi-calculus in differential interaction nets. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *CONCUR*, volume 4703 of *Lecture Notes in Computer Science*, pages 333–348. Springer, 2007.
- [ER03] Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theor. Comput. Sci.*, 309(1):1–41, 2003.
- [ER06a] Thomas Ehrhard and Laurent Regnier. Böhm trees, Krivine's machine and the Taylor expansion of lambda-terms. In Arnold Beckmann, Ulrich Berger, Benedikt Löwe, and John V. Tucker, editors, *CiE*, volume 3988 of *Lecture Notes in Computer Science*, pages 186–197. Springer, 2006.
- [ER06b] Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theor. Comput. Sci.*, 364(2):166–195, 2006.
- [ER08] Thomas Ehrhard and Laurent Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. *Theor. Comput. Sci.*, 403(2-3):347–372, 2008.

- [Gen67] Gerhard Gentzen. *The Collected Works of G. Gentzen*. North-Holland, 1967.
- [Gir87] Jean-Yves Girard. Linear logic. *Th. Comp. Sc.*, 50:1–102, 1987.
- [Gir91a] Jean-Yves Girard. A new constructive logic: classical logic. *Mathematical Structures in Computer Science*, 1(3):255–296, 1991.
- [Gir91b] Jean-Yves Girard. Quantifiers in linear logic II. In Corsi and Sambin, editors, *Nuovi problemi della logica e della filosofia della scienza*, pages 79–90, Bologna, 1991. CLUEB.
- [Gir95] Jean-Yves Girard. Linear logic: its syntax and semantics. In Girard et al. [GLR95], pages 1–42.
- [Gir96] Jean-Yves Girard. Proof-nets: the parallel syntax for proof-theory. In *Logic and Algebra*, volume 180 of *Lecture Notes in Pure and Appl. Math.*, pages 97–124, 1996.
- [Gir98] Jean-Yves Girard. Light linear logic. *Information and Computation*, 143(2):175–204, June 1998.
- [Gir99] Jean-Yves Girard. On the meaning of logical rules I: syntax vs. semantics. In Ulrich Berger and Helmut Schwichtenberg, editors, *Computational Logic*, pages 215–272. Springer, 1999. NATO series F 165.
- [GLR95] Jean-Yves Girard, Yves Lafont, and Laurent Regnier, editors. *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1995.
- [How80] William Alvin Howard. *The formulae-as-types notion of construction*, volume to H.B. Curry: *Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980.
- [Hue80] Gérard P. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *J. ACM*, 27(4):797–821, 1980.
- [HvG03] Dominic Hughes and Rob van Glabbeek. Proof nets for unit-free multiplicative-additive linear logic. In *LICS*, pages 1–10. IEEE Computer Society Press, 2003.
- [Joi93] J.-B. Joinet. *Étude de la Normalisation du Calcul des Séquents Classique à Travers la Logique Linéaire*. Thèse de doctorat, University of Paris VII, 1993.
- [Kfo00] Assaf J. Kfoury. A linearization of the lambda-calculus and consequences. *Journal of Logic and Computation*, 10(3):411–436, 2000.
- [KP90] Jean-Louis Krivine and Michel Parigot. Programming with proofs. *J. Inf. Process. Cybern.*, 26(3):149–167, 1990.
- [Kri93] Jean-Louis Krivine. *Lambda-calculus, types and models*. Ellis Horwood, 1993.

- [Laf90] Yves Lafont. Interaction nets. In *POPL '90: Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 95–108, New York, NY, USA, 1990. ACM.
- [Laf95] Yves Lafont. From proof nets to interaction nets. In Girard et al. [[GLR95](#)], pages 225–247.
- [Lam92] François Lamarche. Sequentiality, games and linear logic (announcement). In *Workshop on Categorical Logic in Computer Science*. Publications of the Computer Science Department of Aarhus University, DAIMI PB-397-II, 1992.
- [Lau99] Olivier Laurent. Polarized proof-nets: proof-nets for LC (extended abstract). In Jean-Yves Girard, editor, *Typed Lambda Calculi and Applications '99*, volume 1581 of *Lecture Notes in Computer Science*, pages 213–227. Springer, April 1999.
- [Lau02] Olivier Laurent. *Étude de la polarisation en logique*. Thèse de doctorat, Université Aix-Marseille II, March 2002.
- [LQTdF05] Olivier Laurent, Myriam Quatrini, and Lorenzo Tortora de Falco. Polarized and focalized linear and classical proofs. *Annals of Pure and Applied Logic*, 134(2–3):217–264, July 2005.
- [LTdF04] Olivier Laurent and Lorenzo Tortora de Falco. Slicing polarized additive normalization. In *Linear Logic in Computer Science*, pages 247–282, 2004.
- [Maz06] Damiano Mazza. *Interaction Nets: Semantics and Concurrent Extensions*. Ph.D. Thesis, Université de la Méditerranée/Università degli Studi Roma Tre, 2006.
- [MM08] Paul-André Melliès and Samuel Mimram. Asynchronous games without alternation. Submitted for publication, 2008.
- [Ohl98] Enno Ohlebusch. Church-Rosser theorems for abstract reduction modulo an equivalence relation. In *Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 17–31. Springer, 1998.
- [Pag06a] Michele Pagani. Acyclicity and coherence in multiplicative and exponential linear logic. volume 4207 of *Lecture Notes in Comput. Sci.*, pages 531–545, 2006.
- [Pag06b] Michele Pagani. *Proof nets and cliques: towards the understanding of analytical proofs*. PhD thesis, Università Roma Tre / Université Aix-Marseille II, April 2006.
- [Pag08] Michele Pagani. Visible acyclic nets: between interaction and semantics. In preparation, 2008.
- [Pag09] Michele Pagani. The cut-elimination theorem for differential nets with boxes. Submitted for publication, 2009.

- [Pao06] Luca Paolini. A stable programming language. *Inf. Comput.*, 204(3):339–375, 2006.
- [Plo77] Gordon D. Plotkin. LCF considered as a programming language. *Theor. Comput. Sci.*, 5(3):225–255, 1977.
- [PTdF08] Michele Pagani and Lorenzo Tortora de Falco. Strong normalization property for second order linear logic. To appear on *Theor. Comput. Sci.*, 2008.
- [Reg92] Laurent Regnier. *Lambda-Calcul et Réseaux*. Thèse de doctorat, Université Paris VII, 1992.
- [Reg94] Laurent Regnier. Une équivalence sur les lambda-termes. *Th. Comp. Sc.*, 126:281–292, 1994.
- [Ret97] Christian Retoré. A semantic characterisation of the correctness of a proof net. *Mathematical Structures in Computer Science*, 7(5):445–452, October 1997.
- [Sch65] David Edward Schroer. *The Church Rosser Theorem*. PhD thesis, Cornell University, Ithaca N.Y., 1965. Ph.D. thesis.
- [Sco72] Dana Scott. Continuous lattices. In Lawvere, editor, *Toposes, Algebraic Geometry and Logic*, volume 274 of *Lecture Notes in Mathematics*, pages 97–136. Springer, 1972.
- [TdF00] Lorenzo Tortora de Falco. *Réseaux, cohérence et expériences obsessionnelles*. Thèse de doctorat, Université Paris VII, January 2000.
- [Ter03] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.
- [Tra08a] Paolo Tranquilli. A characterization of hypercoherent semantic correctness in multiplicative additive linear logic. In Michael Kaminski and Simone Martini, editors, *CSL*, volume 5213 of *Lecture Notes in Computer Science*, pages 246–261. Springer, 2008.
- [Tra08b] Paolo Tranquilli. Intuitionistic differential nets and lambda calculus. Conditionally accepted to *Theor. Comput. Sci.*, 2008.
- [Vau07] Lionel Vaux. *λ -calcul différentiel et logique classique : interactions calculatoires*. Thèse de doctorat, Université de la Méditerranée, 2007.
- [Vau08] Lionel Vaux. Differential linear logic and polarization. Submitted for publication, 2008.

Reference Figures

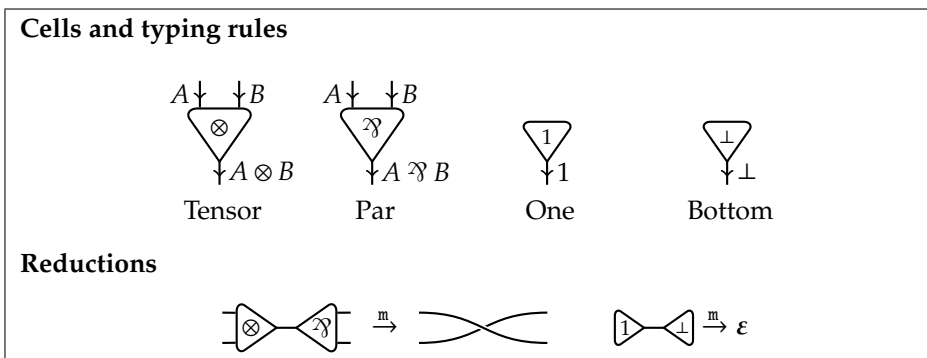


Figure 3.1: Cells, typings and reductions of MLL.

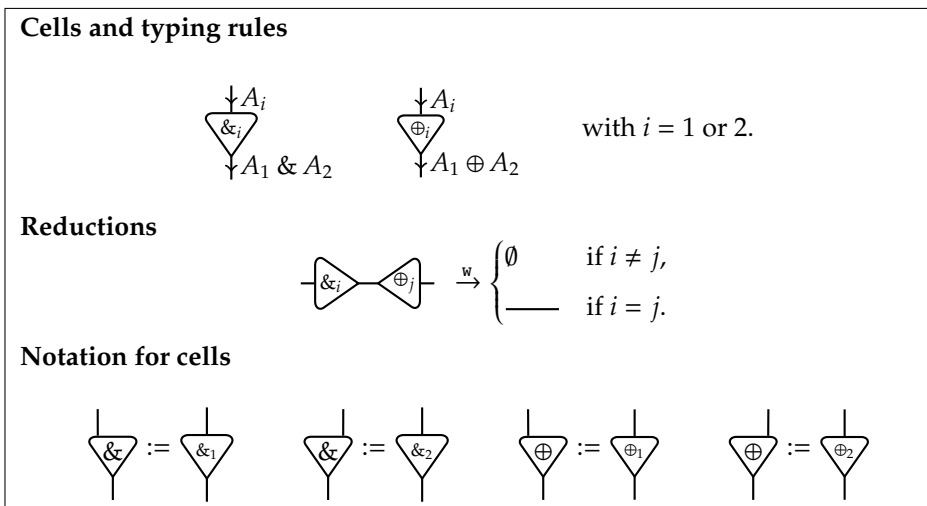


Figure 3.4: Cells, typing rules and reductions of MALL nets. We also show the alternative and intuitive notation, where the symbols $\&_1$ and $\&_2$ (resp. \oplus_1 and \oplus_2) are identified, but recovered from the position of the auxiliary port: left for $\&_1$ (resp. \oplus_1) and right for $\&_2$ (resp. \oplus_2).

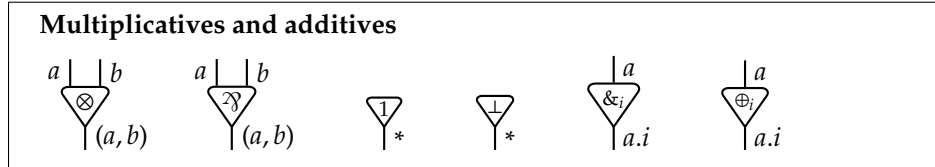


Figure 3.8: Rules for the experiments for all the cells of MALL.

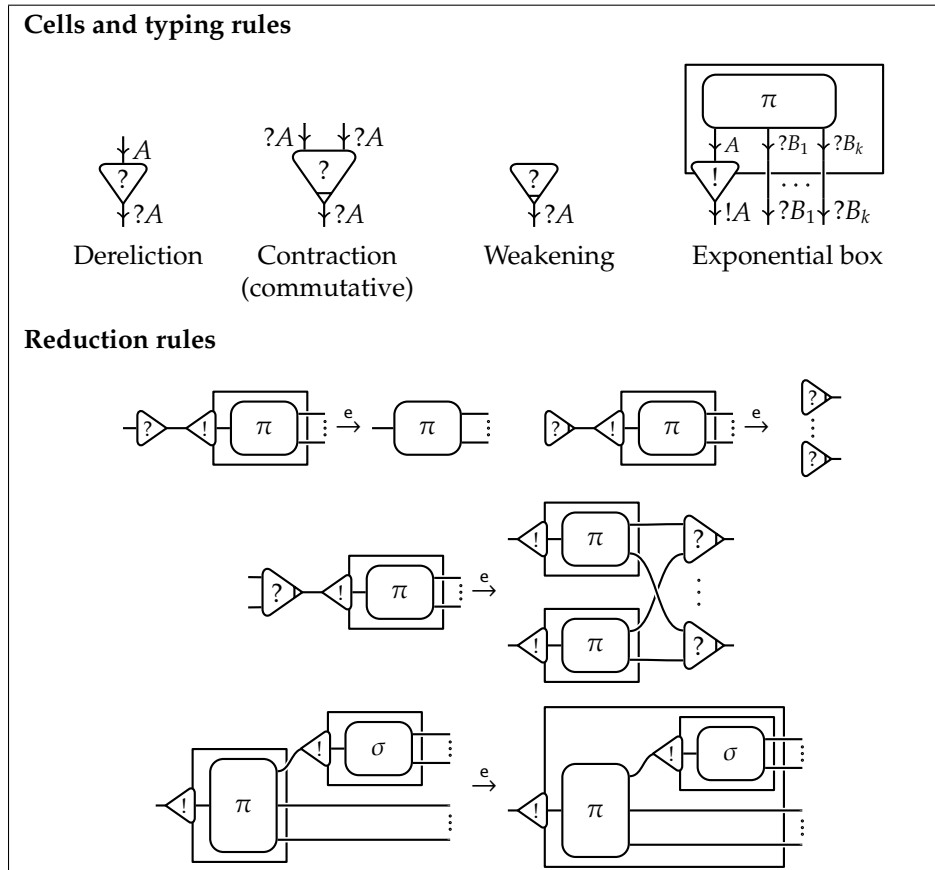


Figure 4.1: Cells of MELL, together with typing rules for cells and for the box and the reduction rules. Recall that in the box on box reduction the two boxes must be different (a condition that is not trivial in incorrect MELL nets).

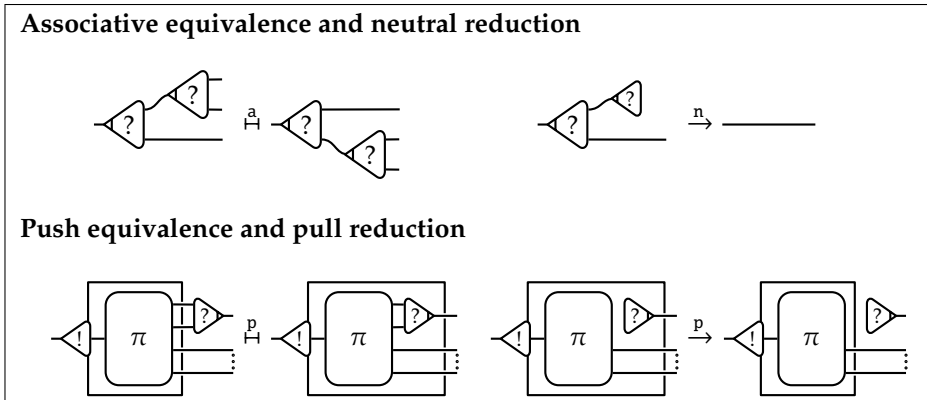


Figure 4.3: The generating pairs for associative and push equivalences, and the neutral and pull reductions.

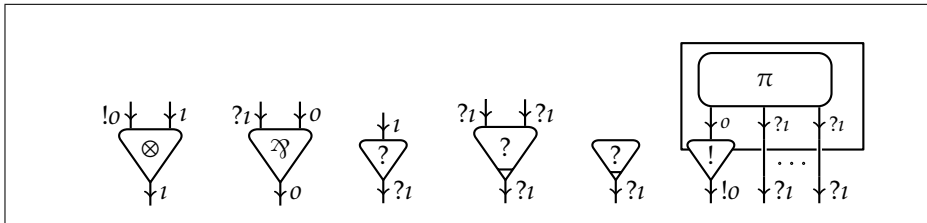


Figure 4.5: Typing rules for pure λ MELL.

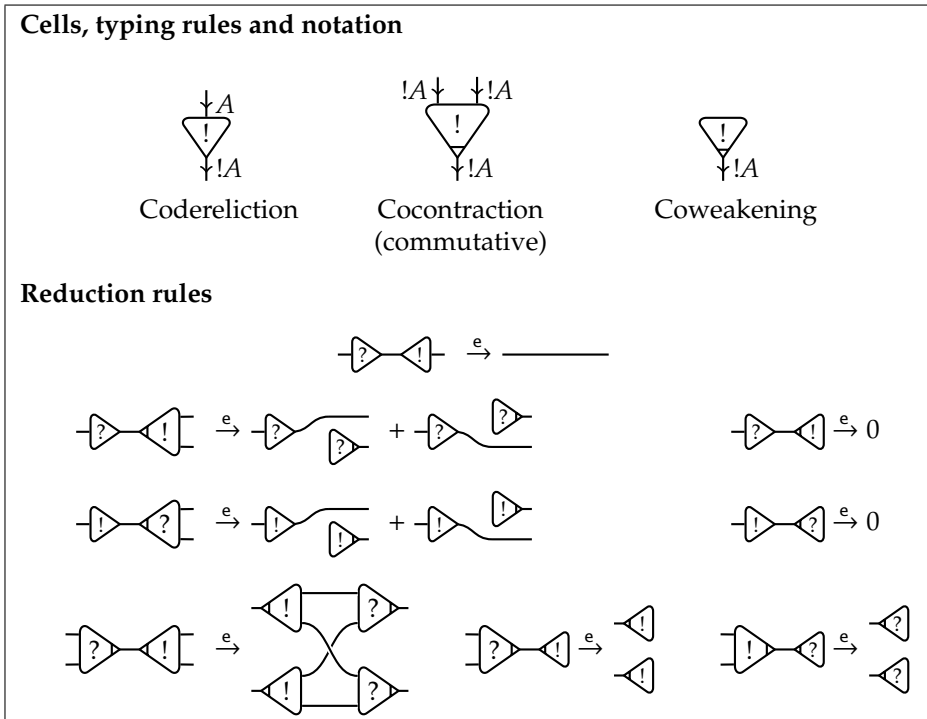


Figure 4.7: The additional cells, typing and reduction rules for differential interaction nets, DiLL_0 . Rules for $!$ and $?$ are totally symmetric.

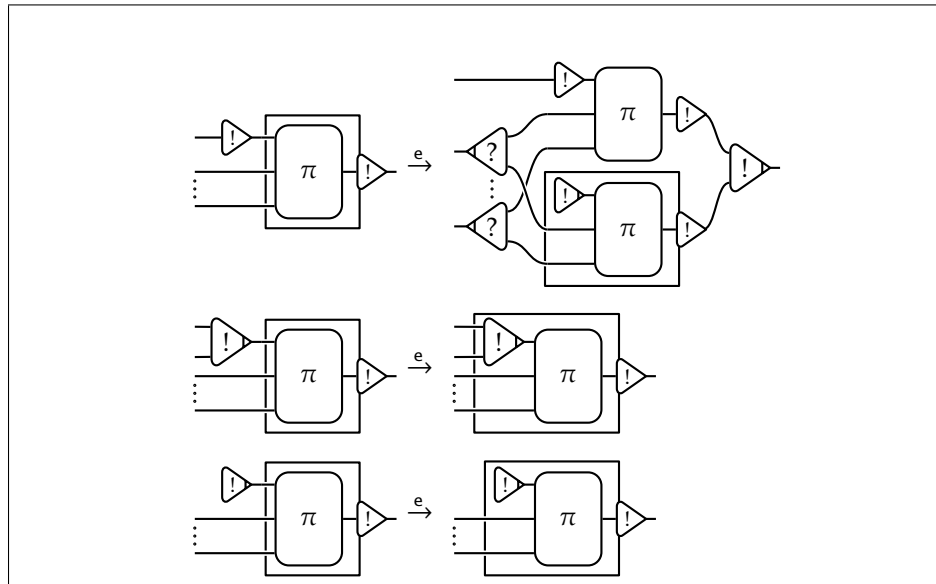


Figure 4.9: Additional exponential reduction rules for DiLL.

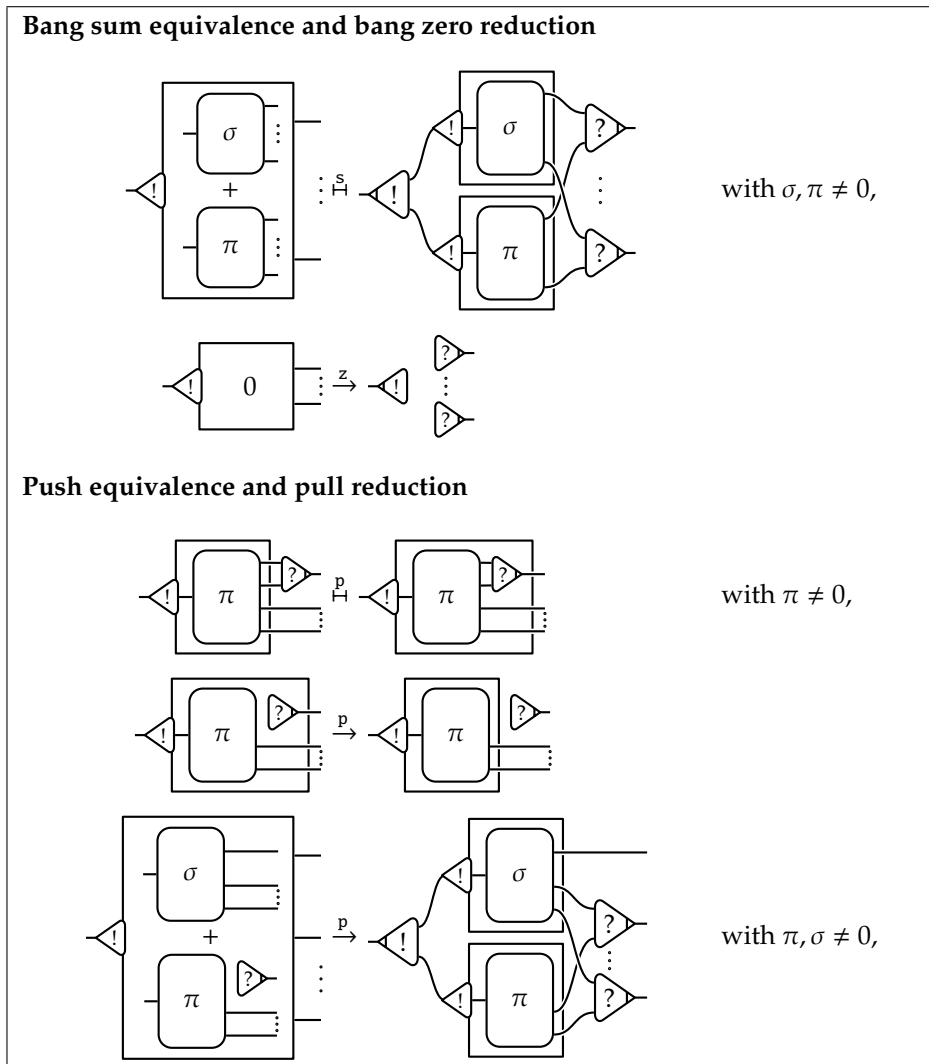


Figure 4.10: The pairs generating the push and sum equivalences, and the pull and zero reductions.

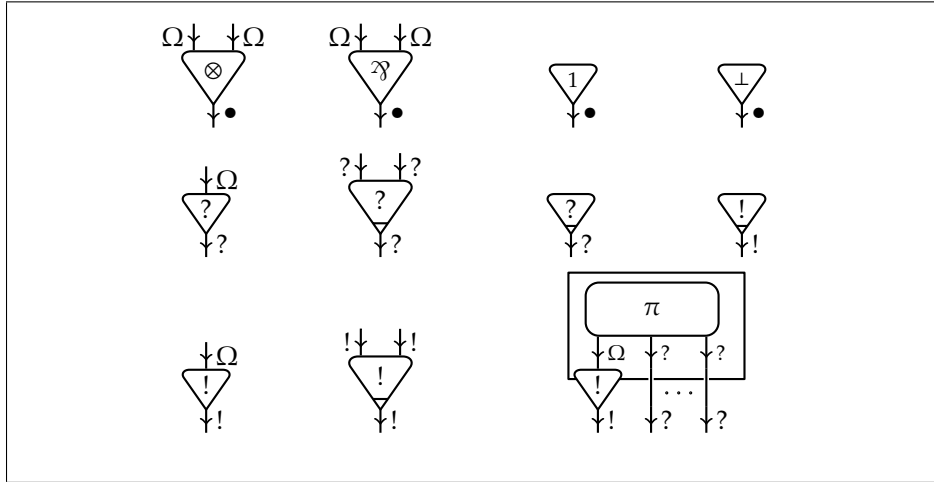


Figure 4.13: Rules for lax typing.

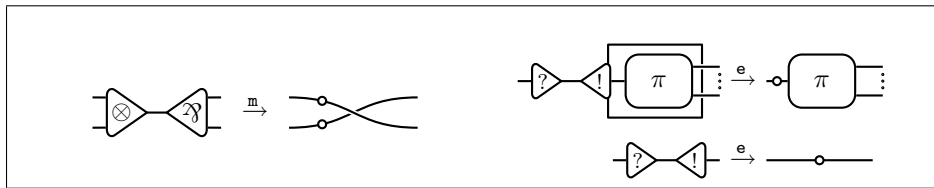


Figure 6.1: The marking rules of DiLL° .




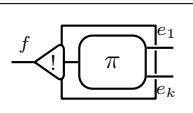
	$?(e) = ?(p)$, the variable associated with the free port p ;
	$?(e) = ?(e_1) + ?(e_2)$;
	$?(e) = ?(f)$;
	$?(e_i) = \begin{cases} ?(f)(1 + \sum_j !(e_j)) & \text{if } \pi = 0, \\ ?(f)(1 + \sum_j !(e_j)) \sum_{\lambda \in \pi} \text{sp}(\lambda) ?(e^\lambda) & \text{otherwise;} \end{cases}$
otherwise:	$?(e) = 1$.

Table 6.1: Rules for the $?$ -weight. $?(e) = 1$ if e is not exponential.

$p \bullet$	$!(e) = \begin{cases} !(p) & \text{if } p \in \text{fp}_0(\pi), \\ !(f) & \text{if } p \in \text{fp}_0(\sigma(B)) \text{ for a box } B, p \text{ is above} \\ & \text{an auxiliary port, and } f \text{ is the wire cor-} \\ & \text{responding to } p \text{ outside the box,} \\ 1 & \text{if } p \in \text{fp}_0(\sigma(B)) \text{ for a box } B \text{ and } p \text{ is} \\ & \text{above the principal port;} \end{cases}$
	$!(e) = !(e_1) + !(e_2);$
	$!(e) = !(f);$
	$!(e) = \begin{cases} 1 + \sum_j !(f_j) & \text{if } \pi = 0, \\ (1 + \sum_i !(f_i)) \sum_{\lambda \in \pi} \text{sp}(\lambda) & \text{otherwise.} \end{cases}$
otherwise:	$!(e) = 1.$

Table 6.2: Rules for the !-weight. $!(e) = 1$ if e is not exponential.

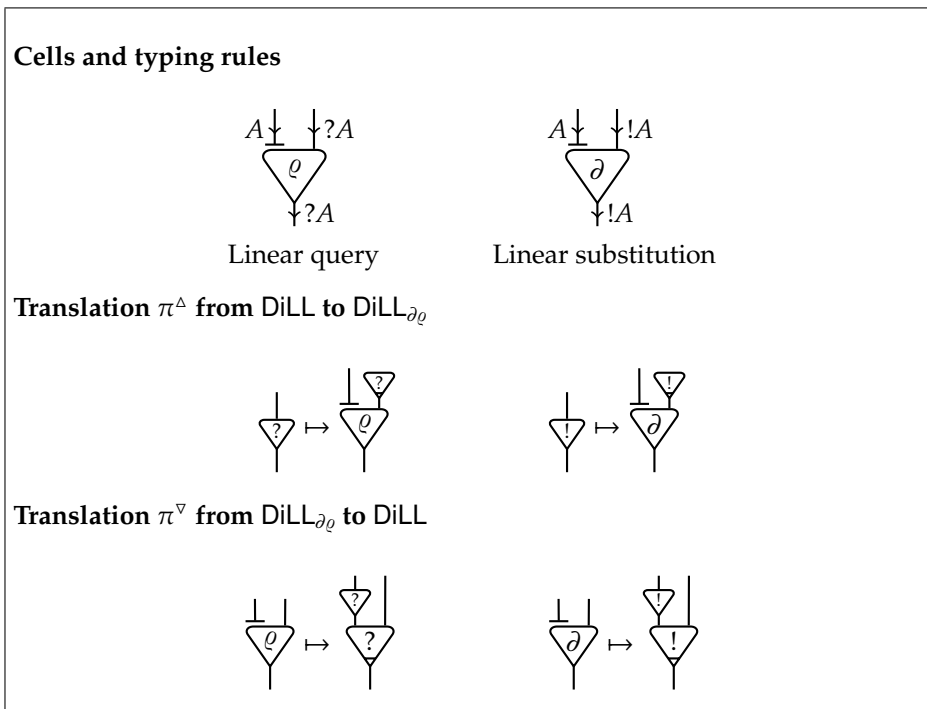


Figure 7.1: The cells ∂ and ϱ of DiLL $_{\partial\varrho}$, and the translation from and to DiLL. The cells are *not* commutative, but may be drawn with swapped inactive ports, hence the special marking of the left port. ϱ is switching, ∂ is not.

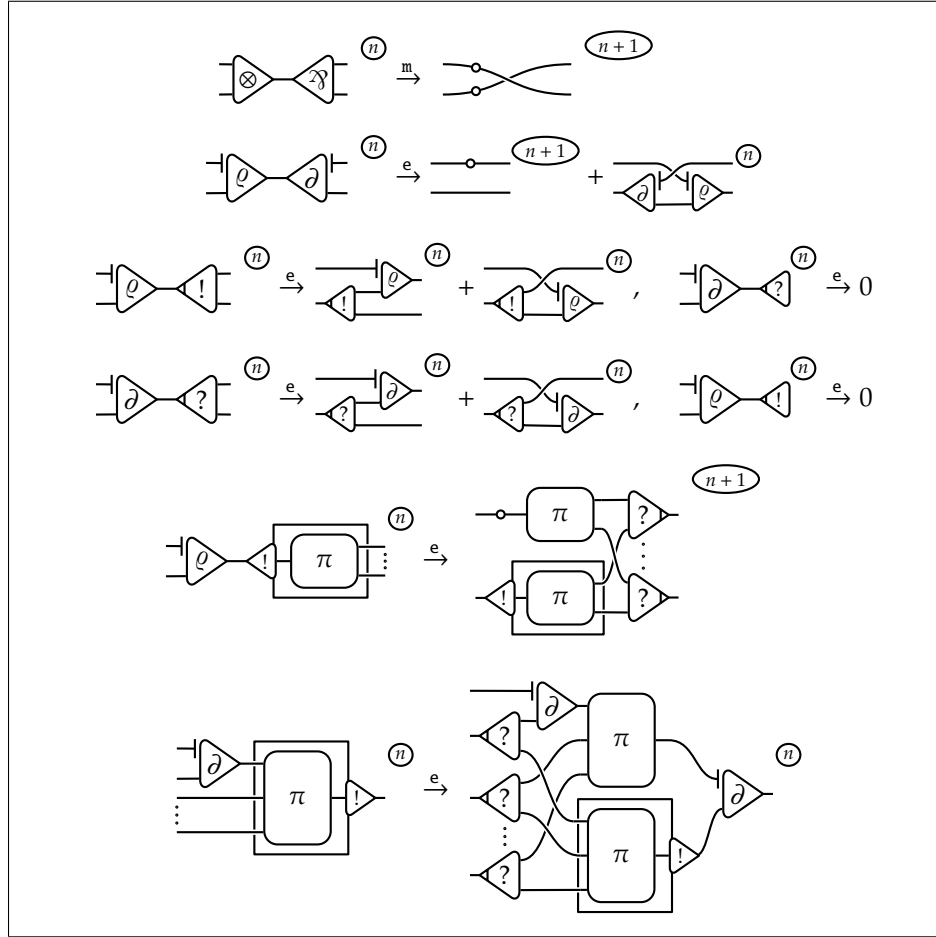


Figure 7.2: Reduction rules of labeled $\text{DiLL}_{\partial\varrho}^{\circ}$. All combinations not shown are the same as in DiLL , with labels left unchanged.

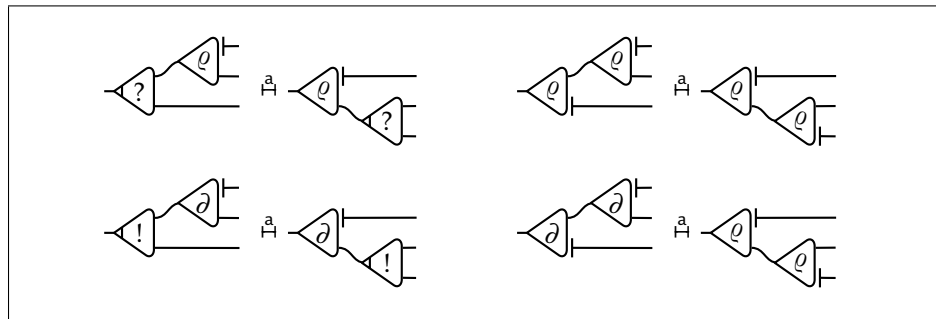


Figure 7.3: The new associative equivalences.

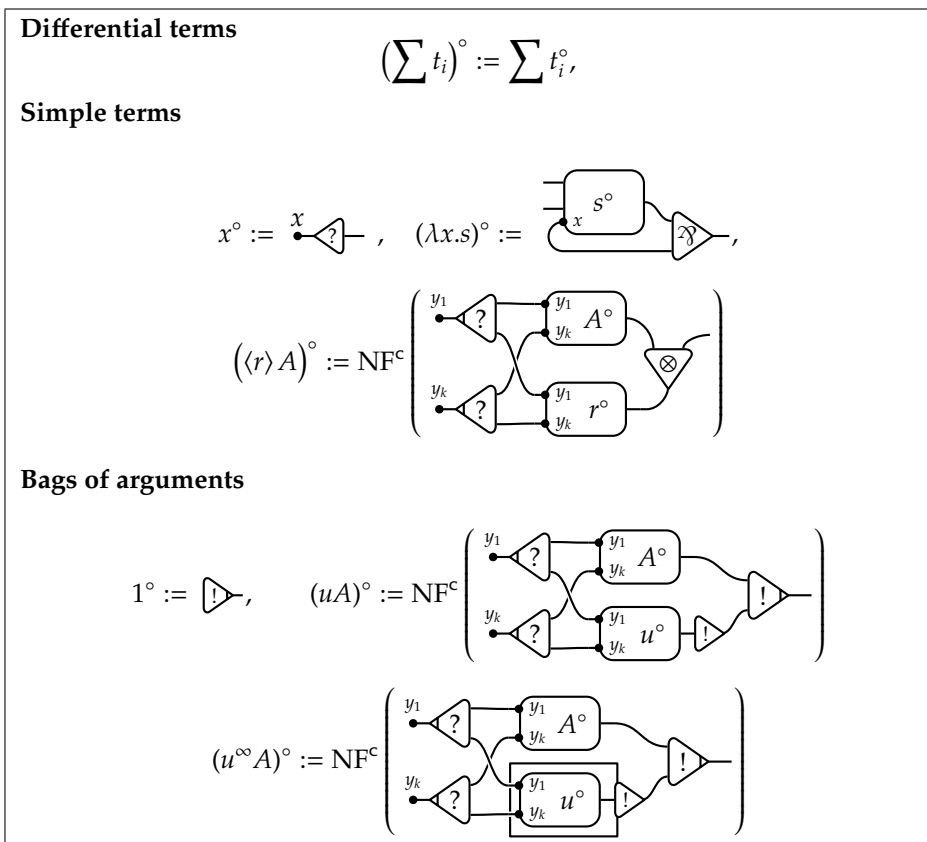


Figure 8.1: Inductive rules for the definition of t° for full resource calculus.