Under consideration for publication in Math. Struct. in Comp. Science

The Conservation Theorem for Differential Nets

MICHELE PAGANI[1] and PAOLO TRANQUILLI[2]

[1] *Dipartimento di Informatica, Università di Torino.*
[2] *LIP, ENS Lyon, Université de Lyon (UMR 5668 CNRS ENS Lyon UCBL INRIA).*

Received December 15, 2009

We prove the conservation theorem for differential nets — the graph-theoretical syntax of the differential extension of Linear Logic (Ehrhard and Regnier's DiLL). The conservation theorem states that the property of having infinite reductions (here infinite chains of cut elimination steps) is preserved by non-erasing steps. This turns the quest for strong normalization into one for non-erasing weak normalization, and indeed we use this result to prove strong normalization of simply typed DiLL (with promotion).
Along the way to the theorem we achieve a number of additional results having their own interest, such as a standardization theorem and a slightly modified system of nets, $\mathsf{DiLL}_{\partial!}$.

1. Introduction

In λ-calculus a β-reduction is *non-erasing* whenever it does not erase the argument involved in the redex, i.e. whenever it results in the substitution of a variable actually occurring in the body of the function. The conservation theorem states that non-erasing β-reduction preserves the property of having infinite reductions (Theorem 13.4.12, p. 339 of [Bar81]). In this paper we prove an analogous theorem for *differential nets* – the graph-theoretical syntax of the differential extension of Linear Logic (DiLL [ER06]).

The conservation property is a powerful tool in the pursuit of strong normalization, as the existence of a maximal non-erasing reduction sequence entails that *all* non-erasing reduction sequences are finite. Moreover, since erasing reductions are obviously finite and can be postponed after the non-erasing ones, a term having a maximal non-erasing reduction sequence enjoys strong normalization for the whole reduction. Indeed another way of wording the theorem is to say that non-erasing reduction is *perpetual*, i.e. when starting from a non SN term non-erasing steps will necessarily go on forever. One can say that the conservation theorem turns the quest for Strong Normalization (SN) into one for non-erasing Weak Normalization (WN). Indeed a wealth of SN proofs for various typed calculi have been achieved through conservation theorems, e.g. [Gan80, Ned73, Sør97].

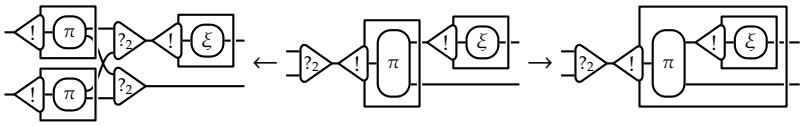
Various techniques have been proposed for achieving SN, alternative to the ones passing through a conservation theorem. Among them let us mention the proofs based on inductions on a suitable ordinal and the ones based on notions of computability or of reducibility candidates. The former usually consists in defining a measure on terms which

decreases at any reduction step, with respect to a suitable well-ordering. Such a proof is worthwhile since it gives an appraisal of the complexity of the reduction by producing explicitly an ordinal on which the induction relies. However it is quite hard to show a decreasing measure even for rather simple calculi: for example at the time of writing no proof of this kind has been given for the simply typed λ-calculus, as far as we know.

On the other hand, the techniques based on computability (e.g. [Tai67]) or on reducibility candidates (e.g. [Gir72]) are very versatile and have been applied to various calculi, even quite complex such as Girard's System F. As Girard underlines in [GLT89], such a versatility depends on the fact that the complexity of the reduction can be kept implicit, "hidden" under the logical complexity of the definition of computability/reducibility candidate (basically an alternation of the quantifiers $\forall$ and $\exists$, as deep as the order of the type assigned to the term). Indeed, there is a common belief that such a logical complexity makes these techniques adaptable enough to be able to turn a proof of WN into one of SN (when it holds) just by minor changes in the definition of computability/reducibility candidate, namely without the need of some additional, complex lemmata such as a conservation theorem.

Although this is unquestionable for typing systems of the λ-calculus or similar calculi thereof, it might not arguably be the case in general. It seems that achieving SN by means of computability/reducibility candidates may at times need nevertheless a conservation property. For example, in some typed λ-calculi with explicit substitution, applying the technique of reducibility candidates to achieve SN needs some additional lemmata stating a property similar to the conservation one. This might be due to the fact that systems with explicit substitutions often are not (even weakly) orthogonal, in the sense that they have reduction rules overlapping and leading to non-trivial critical pairs.

The proof nets of Linear Logic (LL [Gir87a]) yield another example of non orthogonal rewriting system for which the reducibility candidates technique needs a conservation theorem to achieve SN. Proof nets are graph-theoretical structures representing proofs of LL, both quotienting on unessential rule commutations and providing more local cut-elimination rules. On the one hand proof nets avoid several commutative steps needed in sequent calculus instead, cutting on the bureaucracy needed to ready the cuts to be fired. On the other hand their cut-elimination is more local than in natural deduction, dividing in smaller steps the substitution of hypotheses. Proof nets yield a non orthogonal rewriting since they have non-trivial critical pairs, as in



where we have two redexes overlapping at a box $!\pi$ (which represents the promotion rule of LL applied to a proof net π): one with a contraction $?_2$ and the other with another box $!\xi$, leading to different reductions. Obviously the reader unfamiliar with proof nets will find the definitions in section 2.

As hinted above, though the reducibility candidate method can be applied to LL proof nets, the further ingredient of a conservation theorem is needed. In the original paper

by Girard such theorem is given the name of *standardization*[1] (theorem 4.25 p. 72 of [Gir87a]), however its statement is incomplete and its proof is omitted altogether (see the discussion in [PTdF10]). Danos gives a satisfactory proof of it for MELL, the multiplicative exponential fragment of LL, baptizing the result as *striction lemma* (théorème 8.31 p. 64 of [Dan90]). Finally the first author and Tortora de Falco show it in [PTdF10], where it dons again Girard's name: the result is for a syntax for full LL (*sliced* proof nets) which is quite liberal with respect to the representation of the additive structure. Again, such theorem is used to infer the SN property from the WN one obtained by reducibility candidates, leading in turn to SN of full second order LL proof nets of [Gir87a] via a simulation in sliced proof nets. It must be noted though that the conservation theorem is not directly obtained for the LL proof nets of [Gir87a].

Here, apart from turning again to the *conservation* nomenclature, we concentrate over the syntax of differential linear logic, DiLL's *differential nets* [ER06], and prove the conservation theorem in this system. DiLL extends MELL both with new rules (*codereliction*, *cocontraction* and *coweakening*, symmetric to the correspondingly named rules of MELL) and with a formal sum modelling a form of internal nondeterminism, syntactically somewhat similar to how slices are aggregated in sliced proof nets. In order to properly state the theorem we need to precisely define what a non-erasing reduction is (Definition 2): in fact reductions to 0 introduce new ways to erase parts of the nets, with the particularity of extending their erasing capabilities outside of the redex originating them.

As already explained, the conservation theorem will enable to transfer results of WN to SN, notably for simply typed and second order DiLL. While we put the latter aside, we will show how this is carried on for the former at the end of the paper (section 5), by integrating the proof of WN given by the first author in [Pag09] with the finite development theorem given by the second one in [Tra09a].

The interest in DiLL lies in several aspects. First, it is based on a solid and interesting semantics, namely *finiteness spaces* [Ehr05]. These domains arise by relaxing the notion of duality proper to coherent spaces (the archetypal semantics of LL [Gir87a]). However they give rise to a novel interpretation of LL as they can be equivalently seen as special kinds of topological vector spaces, where all the operations of LL are mapped to the corresponding operations of linear algebra. Most importantly, the usual factorization of the intuitionistic arrow/function space $A \to B \cong !A \multimap B$ gets inflected as the space of *analytic functions*. Namely, finiteness spaces allow for a notion of differentiation, which were then lifted to the syntax as shown in [ER03] (for λ-calculus) and [ER06] (for LL, giving rise to the above mentioned symmetrization of its rules).

Moreover, though finiteness spaces and DiLL allow for a sort of tamed nondeterminism, differential nets turn out to be a well-behaved rewriting system. After the first results stated in [ER06] but restricted to the promotion-free fragment, other works have strengthened this claim: as already mentioned, the first author showed WN of simply typed DiLL in [Pag09], the second showed confluence (or rather Church-Rosser modulo, as we will have occasion to explain) of the untyped case in [Tra09a], and we here add yet

[1] Such nomenclature is quite misleading when related to λ-calculus. We reserve the name "standardization" for a result (Theorem 21) much closer to the renowned standardization theorem of λ-calculus.

another brick to the solidity of the system. Confluence in presence of a formal sum modeling nondeterminism means that such a nondeterministic choice is completely internal and does not depend on the reduction strategy.

Finally, an enticing point of interest is the link with concurrency. Differentiation, as modeled by the codereliction rule, can be in fact seen as the introduction of a one-use entity that goes to only one among the possibly many who are asking for it. This has been employed by Ehrhard and Laurent [EL07] to model concurrent communication, where only one among many may answer to a given signal. Their work establishes then a translation from a finitary fragment of π-calculus into (essentially) promotion-free differential nets, backed by a bisimulation. In this way a simple yet profound mathematical concept such as differentiation is linked to an important computer science one such as inter-process concurrent communication.

As for [PTdF10], our proof of the conservation theorem relies on what Girard refers to as Gandy's method ([Gir87b], referring to [Gan80]), while called Nederpelt lemma by Bezem and Klop ([Ter03], referring to [Ned73]). Namely, we aim at defining an increasing measure which together with some form of confluence yields the implication $\mathsf{WN} \implies \mathsf{SN}$, via Gandy/Nederpelt lemma. We give an updated version of this result in the case of reduction modulo (Lemma 37 in Appendix A), as equivalences were already shown to be necessary in [Tra09a]. As we will see confluence of non-erasing reduction fails in DiLL, so we will pass by a slightly modified system enjoying it, $\mathsf{DiLL}_{\partial\varrho}$. Once we will have defined DiLL in detail, we will spend some more words on the difficulties and the ideas motivating this choice (see subsection 2.3).

Outline. In section 2 we will introduce all the syntax on which this paper is based. After a general introduction to the interaction net paradigm, both DiLL and $\mathsf{DiLL}_{\partial\varrho}$ are defined, together with the notion of non-erasing reduction. The geometric correctness criterion of switching acyclicity is also presented, which is necessary for most of the proofs to work.

Section 3 is devoted to proving the conservation theorem for $\mathsf{DiLL}_{\partial\varrho}$, which is followed in section 4 by the main result of the paper, the same property for DiLL.

Section 5 gives an example of the application of such theorem, by showing how SN of simply typed DiLL can be deduced, after defining what a simple typing is.

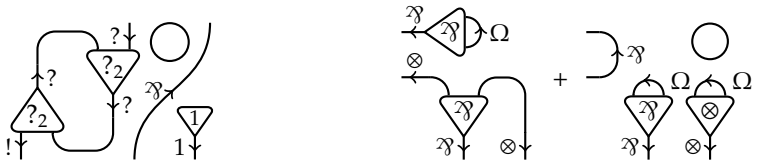
Finally Appendix A gives the necessary notions of rewriting and rewriting modulo an equivalence, and states the lemmata needed in the rest of the paper, proving those which were not found in the literature.

2. Differential Nets

2.1. *Preliminaries*

Differential nets are defined on top of Lafont's interaction nets [Laf90], which intuitively are circuits made of cells and wires connecting their ports. As we will see in subsection 2.3, the conservation theorem fails when considering truly untyped nets, hence we will deal with a very weak form of typing.

Suppose given an alphabet Σ of symbols and a set $\mathcal{T}$ of types endowed with an invo-



(a) a simple net (b) a polynet

Figure 1. Example of a simple net and of a polynet with two addends.

lutive negation $(.)^\perp : \mathcal{T} \to \mathcal{T}$. A **typed interaction net** α is the given by the following data.

— A set $\mathsf{p}(\alpha)$ of **ports**.

— A set $\mathsf{c}(\alpha)$ of **cells**; to each cell c is assigned a **symbol** $\sigma(c)$ in Σ, a port in $\mathsf{p}(\alpha)$ called **principal**, and a number of other, **auxiliary** ones. How the latter are treated distinguishes between two kinds of cells: in **non commutative** ones, auxiliary ports are a finite sequence, in **commutative** ones they form a finite set. We denote by $\mathsf{deg}(c)$ the number of ports (principal and auxiliary) of the cell c, which is determined by the symbol $\sigma(c)$. Every port in $\mathsf{p}(\alpha)$ can be associated with at most one cell; a port associated with a cell is called **connected**, otherwise it is **free**. Free ports (also called conclusions) are denoted by $\mathsf{fp}(\alpha)$.

— A set $\mathsf{w}(\alpha)$ of **wires**, which can be either unordered pairs $\{p, q\}$ of ports, or **loops**, i.e. wires not connecting any port (intuitively short circuited wires). Each port is in exactly one wire. **Directed wires**, denoted by $\vec{\mathsf{w}}(\alpha)$, are ordered pairs (p, q) such that $\{p, q\}$ is a wire; an involutive duality is trivially defined by $(p, q)^\perp := (q, p)$. **Terminal wires** are the directed ones going to the free ports.

— A function $\ell : \vec{\mathsf{w}}(\alpha) \to \mathcal{T}$ (the **typing**) preserving duality, i.e. such that $\ell(d^\perp) = \ell(d)^\perp$. Without going too much into details, a typing discipline is enforced by imposing conditions which depend on the symbols of the cells.

Notice that a typing can be equivalently defined by assigning types to ports rather than wires, imposing that ports on the two ends of a wire get dual types.

In fact, in what follows we work with graphical representations of nets, which are much more intuitive to deal with. In such representations only cells and wires are drawn, and ports are left implicit, corresponding to wire extremities. A cell is represented as a triangle, with its symbol depicted inside; the principal port is identified with an apex of the triangle while the auxiliary ones lay on the opposite side. Free ports appear as extremities of dangling wires. Types may label wires once directions are chosen by drawing arrows on them. For example, the net in Figure 1(a) has five free ports, two cells of symbol $?_2$, one cell of symbol 1 and one loop. The set of types used in the figure will be defined in a while.

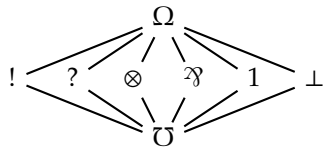
In order to introduce differential nets we also need a notion of addition between nets, intuitively modelling a non-deterministic superposition of nets. Hence we define a **polynet** as a formal finite sum of typed interaction nets, or equivalently a finite multiset of nets, all sharing the same free ports and such that corresponding terminal wires have the same

type. At times we distinguish nets (thus singletons) from polynets by calling them **simple nets**. Figure 1(b) gives an example of polynet with two addends. The correspondence of the free ports of the addends is given by the order of the terminal wires (from left to right and from up to bottom).

Initial Greek letters α, β, γ, δ (resp. middle Greek letters π, ξ, ρ, ν) range over simple nets (resp. polynets). We use additive notation for multisets: 0 is the empty multiset, $\pi + \xi$ is the multiset union of π and ξ adding multiplicities.

2.2. *The System* DiLL

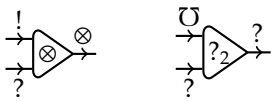
Lax types. We will use the following eight types, called **lax types**: ?, !, $\otimes$, $\parr$, 1, $\bot$, $\mathcal{O}$ (**error**) and Ω (**any**), i.e. the usual 6 type constructors of MELL (though all taken as *constant* types) together with two new ones. The involutive duality of lax types is defined by $!^\bot = ?$, $\otimes^\bot = \parr$, $1^\bot = \bot$ while $\mathcal{O}$ and Ω are autodual. Moreover they are given an order $\sqsubseteq$ defined by just setting Ω and $\mathcal{O}$ as the top and bottom elements respectively, as depicted below.



It is not a subtyping relation as it is preserved, rather than inverted, by duality. Given two lax types A and B the type $A \sqcap B$ is defined as the meet of the two. So for example $? \sqcap ! = \mathcal{O}$ and $\otimes \sqcap \Omega = \otimes$. Notice that $(A \sqcap B)^\bot = A^\bot \sqcap B^\bot$.

Statics. The promotion rule of linear logic corresponds in the syntax of nets to the **exponential box**, which is a special cell having a whole polynet as symbol. This means that the set DiLL of differential (poly)nets should be given by induction on the exponential depth: first we define DiLL_d for every $d \in \mathbb{N}$ and then we set $\mathsf{DiLL} := \bigcup_{d=0}^{\infty} \mathsf{DiLL}_d$.

The nets and polynets of DiLL_0 are built from all the symbols in Figure 2 but the box one. The typing of the ports should respect the rules shown in the figure, which must be intended *downward closed* with respect to $\sqsubseteq$, so that for example



are both admissible typings. Notice in particular that every net is typable in this way, by just assigning $\mathcal{O}$ to all wires. Lax typing will play a real role in the definition of context closure and reduction, as we will see in a while. Apart from types, DiLL_0 gives exactly the differential *interaction* nets presented in [ER06].

The nets and polynets of DiLL_{d+1} are the ones built from the cells of Figure 2 where for each box c its symbol is a polynet $\sum \alpha_i$ in DiLL_d and there is a bijection between the ports of c and the free ports of $\sum \alpha_i$. The symbol $\sum \alpha_i$ of the box c is also called its

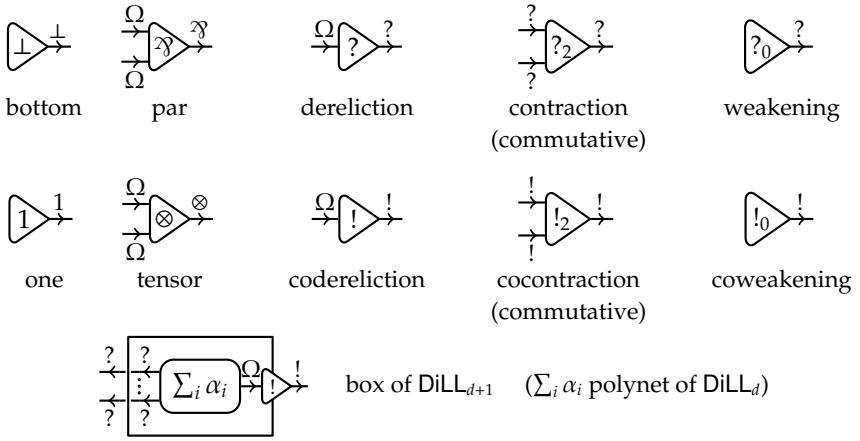
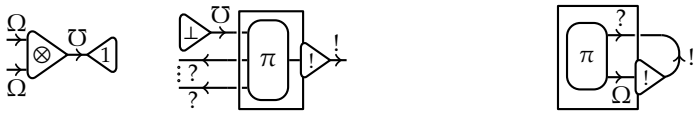


Figure 2. The cells of differential interaction nets, together with their typing rules.



(a) clashes (b) deadlock

Figure 3. Examples of clashes and deadlock.

contents. We will denote by $!(\sum \alpha_i)$ a generic box having $\sum \alpha_i$ as contents. As can be be seen in the figure, boxes follow a special graphical convention with respect to other cells.

A DiLL polynet π is one of DiLL$_d$ for any d; the **depth** of π is the minimal d such that $\pi \in$ DiLL$_d$ (in fact, the maximal number of nested boxes). Let $\mathsf{p}_!(\pi)$, $\mathsf{fp}_!(\pi)$, $\mathsf{c}_!(\pi)$, $\mathsf{b}_!(\pi)$ and $\mathsf{w}_!(\pi)$ be respectively the set of all occurrences of ports, free ports, cells, boxes, and wires occurring in π, including in all contents of the boxes in π. We can partition those sets by depth, so we will denote by $\mathsf{p}_i(\pi)$, $\mathsf{fp}_i(\pi)$, $\mathsf{c}_i(\pi)$, $\mathsf{b}_i(\pi)$ and $\mathsf{w}_i(\pi)$ the corresponding elements contained in i nested boxes, where i is called the depth of the element in π[2].

Dynamics. A port is **active** if it is either a principal one, or an auxiliary one of a box. A wire linking two active ports is a **cut**. A **clash** is a cut that can be typed exclusively with $\mathfrak{O}$, a **deadlock** is a cut between two ports of the same cell (namely a box). In Figure 3 we give examples of deadlock and clash.

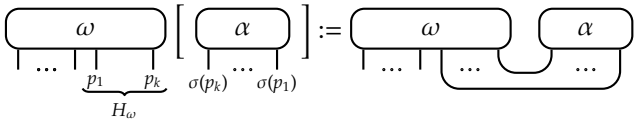
We will now define some reductions on nets, among which cut-elimination is the chief one. Deadlocks and clashes are as usual non-reducible cuts, hence being a normal form does not entail being cut-free. This oddity with respect to a standard proof-theoretical framework is due to the wide freedom we authorize: in fact lax typing, as opposed to

[2] All of this can be defined more formally by an inductive definition. Nevertheless we leave it to the reader as an easy exercise.

usual typing (see section 5), allows clashes, while the absence of a logical correctness criterion allows deadlocks. One may wonder therefore what is the role of lax typing. In fact we set *all* $\mathfrak{O}$-typed cuts to be irreducible, and set in place a mechanism dynamically blocking certain cuts in this way. More details will be given in the following, while the reasons behind lax typing will be given while explaining the example in Figure 8.

As with various calculi, the reduction of differential nets can be presented as the context closure of a set of reduction rules, presented as pairs consisting of a simple net (the **redex**) and a polynet (the **contractum**).

A **linear context** $\omega[\ ]$ is a simple net ω together with a subset H_ω of its free ports (the **hole** of $\omega[\ ]$). The set of the free ports of $\omega[\ ]$ is redefined as $\mathsf{fp}(\omega[\ ]) := \mathsf{fp}(\omega) \setminus H_\omega$. It is linear as it is not a sum and the hole is not inside a box. Given a simple net α and a bijection σ between H_ω and the free ports of α, the plugging $\omega[\alpha]$ of α in the hole of $\omega[\ ]$ amounts to identifying the ports according to σ and welding the wires that come together in this way[3].



The typing of $\omega[\alpha]$ will be obtained from the one of ω and the one of α by setting for each directed wire the type $A_1 \sqcap \cdots \sqcap A_k$ where A_i are all the types of the directed wires that got merged together during gluing. Notice that the typing of $\omega[\alpha]$ respects the condition of assigning dual types to opposite wire directions since $(A \sqcap B)^\perp = A^\perp \sqcap B^\perp$ holds; moreover it respects the typing discipline as typing rules are downward closed for $\sqsubset$. The definition of plugging is then extended by linearity when we plug a polynet, by setting $\omega\left[\sum_i \alpha_i\right] := \sum_i \omega[\alpha_i]$.

Finally, **contexts** generalize the concept in the following way. A linear context $\omega[\ ]$ is a context; furthermore if $\omega[\ ]$ is a context then:

1. $\psi[\omega[\ ]]$ for $\psi[\ ]$ linear is a context[4] (notice $H_{\psi[\omega[\ ]]} = H_\omega$ and $\mathsf{fp}(\psi[\omega[\ ]]) = \mathsf{fp}(\psi[\ ])$);
2. $\omega[\ ] + \pi$ for π polynet is a context (recall that by definition $\mathsf{fp}(\omega[\ ] + \pi) = \mathsf{fp}(\pi) = \mathsf{fp}(\omega[\ ]) = \mathsf{fp}(\omega) \setminus H_\omega$);
3. $!\omega[\ ]$, i.e. a box containing a context, is a context (again by definition the ports of the box will correspond to the free ports of $\omega[\ ]$ as redefined).

The **depth of the hole** of $\omega[\ ]$ is defined as the number of nested boxes containing the hole. Plugging is easily extended to all contexts. In particular notice that sums do not commute with boxes, i.e. $!\left(\omega[\sum_i \alpha_i]\right) = !\left(\sum_i \omega[\alpha_i]\right) \neq \sum_i !(\omega[\alpha_i])$. Final Greek letters $\psi[\ ], \omega[\ ]$ range over contexts. The **context closure** $\tilde{R}$ of a relation R is then defined by $\pi\ \tilde{R}\ \xi$ iff $\pi = \omega[\pi']$, $\xi = \omega[\xi']$ and $\pi'\ R\ \xi'$. In this case, taking $\omega[\ ]$ with maximal hole depth[5] we call the depth of the hole in ω the **depth of the step** $\omega[\pi']\ \tilde{R}\ \omega[\xi']$.

[3] For the quite delicate technical details the reader is referred to [Vau07, dF09].
[4] Composition of contexts should be defined, but it is trivial once plain plugging is defined.
[5] In general the depth may vary, though in the cases of the reduction and conversion steps we will define such a depth will be unique.

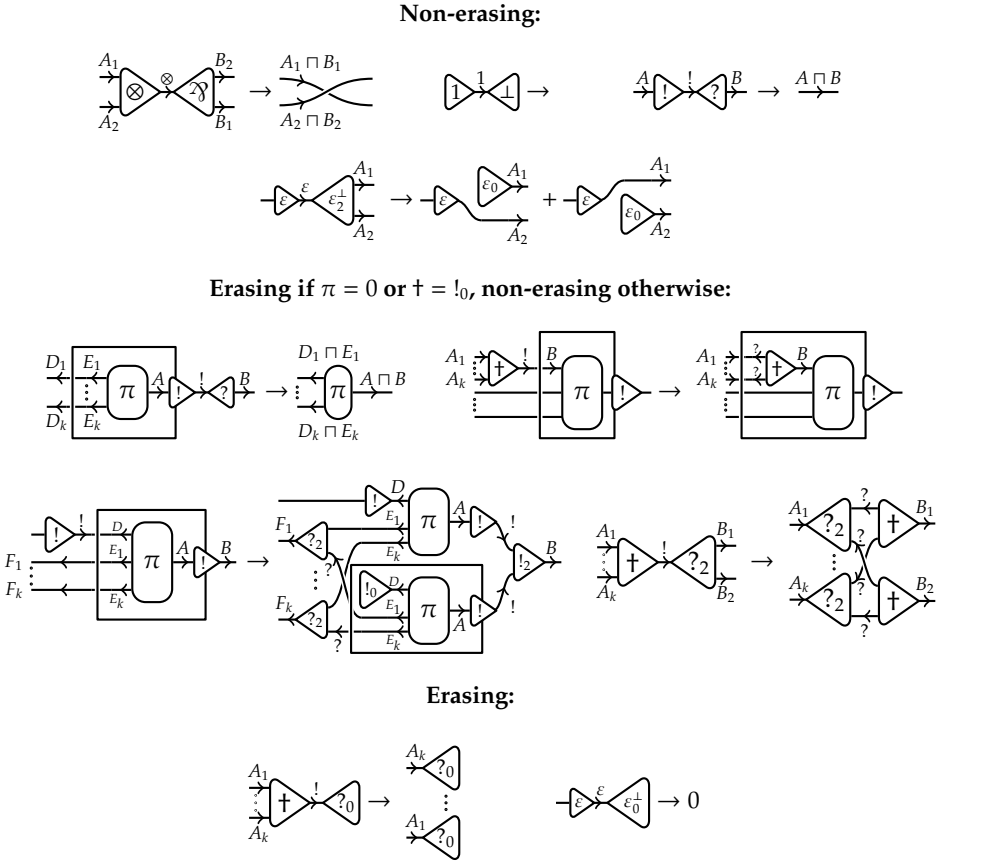


Figure 4. Reductions defining the cut-elimination of DiLL. The symbol † denotes either $!\xi$ (i.e. a box), $!_2$ or $!_0$; the symbol ε denotes either ! or ?. Notice that both 1 on $\perp$ and $?_0$ on $!_0$ reduce to the empty net, not to be confused with 0. Notice that the lax type on the cut is a requirement for the reduction, equivalent to it not being $\mho$. Some lax types are omitted as they do not participate and are unchanged by the reduction.

We are now able to define **cut-elimination** $\xrightarrow{\text{cut}}$ as the context closure of the rules in Figure 4. Notice that each rule fires on a unique cut, and it is identified by the ordered pair of the symbols of the two interacting cells, so e.g. $1/\perp$, $?_0/!_2$, etc. will denote a specific rule. In particular $!\pi/!\xi$ refers to the rule firing on a cut between the principal port of $!\pi$ and an auxiliary port of $!\xi$. We also call such pairs the **kind** of either the reduction rule, the fired cut or the redex.

Remark 1. Let us point out some features specific to lax typing.

1. Lax typing can change during reduction, and its point is exactly that: the aim of lax typing is to memorize those cuts that are or *have been* clashes, using the $\mho$ type. Indeed the condition on the type of the cut of each redex (i.e. not being $\mho$) blocks the reduction of any cut that is a descendant of a clash. This is our reason for including

typing as an integral part of the definition of net in subsection 2.1: lax typing is not a certificate that can be discarded once the type assignment has been done, as it has an active role in defining reduction.
2. Thanks to the fact that typing rules are downward closed with respect to $\sqsubseteq$, it can be seen that each rule preserves the lax typing modulo $\sqsubseteq$.
3. Given a net and a redex in it, due to the gluing operation changing the types on the two interfaces (rather than requiring them to match as is usual with such an operation), there may be different typings given to the redex and the context. However it is straightforward to see that the result of reducing the redex does not depend on such differences (a fact boiling down to associativity of the $\sqcap$ operator).

More discussion on lax typing will be done while discussing Figure 8.

In addition to cut-elimination we consider the **equivalences** and **canonical reductions** of Figure 5. The reason for taking them into consideration is that cut-elimination in DiLL fails to give a confluent rewriting, not even locally. Indeed we cannot ignore associativity of (co)contractions and neutrality of (co)weakening over (co)contraction (see [Tra09a] and the discussion of subsection 2.3). This prompts us to introduce the former as an equivalence and the latter as a reduction. As we need to consider reduction modulo an equivalence anyway, we also study other equivalences (backed by semantic and observational equivalence) which are optional though must be taken together. Each equivalence is accompanied by a reduction which in a sense settles a zeroary case of the equivalence. Reversing each of these gives unwanted looping reductions. The **associative**, **push** and **bang sum** equivalences (denoted by $\overset{\mathtt{a}}{\sim}$, $\overset{\mathtt{p}}{\sim}$ and $\overset{\mathtt{s}}{\sim}$ respectively), together with the **neutral**, **pull** and **bang zero** reductions (which *do not* reduce cuts and are denoted by $\xrightarrow{\mathtt{n}}$, $\xrightarrow{\mathtt{p}}$ and $\xrightarrow{\mathtt{z}}$), are the context closure of the rules shown in Figure 5. The $\pi, \xi \neq 0$ condition is needed, lest one would be able to spawn trees of contractions from nothing, giving dummy infinite reductions.

The push equivalence[6] has already been studied in the literature on proof nets and explicit substitutions [DCG99, DCKP03]. The pull reduction comes here in two versions, the total (pulling away weakenings from inside boxes) and the partial one (which in fact does the same but from just a part of the sum inside, doing a sort of on-the-fly s-conversion). The total pull reduction, which is the one making sense also in LL, was indeed considered in [DCK97, DCG99]. The partial one is needed to give local coherence and is thus necessary to give Church-Rosser modulo (see Appendix A for the notions relative to confluence in presence of an equivalence relation).

It was already known that the bang sum and zero are valid semantically and observationally. As the second author has shown in [Tra09a], we can (and must, if we are to use the push and pull rules) also treat them syntactically. It is interesting to note that these rules implement the well known exponential isomorphisms $!A \otimes !B \cong !(A \,\&\, B)$ and $!\top = 1$ from linear logic (see [Ehr05]).

[6] The name may be misleading, as an equivalence is not directed. It comes from [DCK97] and [Tra08] where it was treated as a reduction, and we felt like keeping it because of the good name pairing with the pull reduction.

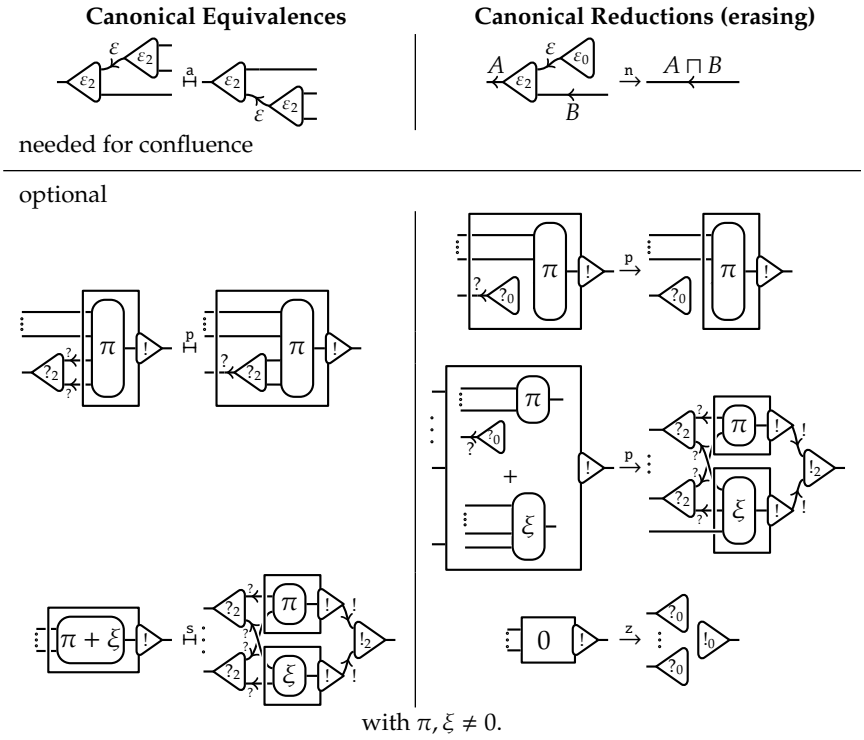


Figure 5. The rules defining equivalences (left) and canonical reductions (right): the symbol $\overset{\mathtt{a}}{\text{H}}$ (resp. $\overset{\mathtt{p}}{\text{H}}$, $\overset{\mathtt{s}}{\text{H}}$) denotes the one-step symmetric conversion generating the associative equivalence $\overset{\mathtt{a}}{\sim}$ (resp. the push one $\overset{\mathtt{p}}{\sim}$ and the bang sum one $\overset{\mathtt{s}}{\sim}$). At the right we have the rules for neutral reduction $\overset{\mathtt{n}}{\to}$, the two pull ones $\overset{\mathtt{p}}{\to}$ (total above and partial below) and the bang zero one $\overset{\mathtt{z}}{\to}$. The condition $\pi, \xi \neq 0$ applies to all rules. The symbol ε denotes either ! or ?. Most of the lax types are omitted as they are left unchanged.

From now on $\sim$ and $\overset{\mathtt{can}}{\longrightarrow}$ (equivalence and canonical reduction, respectively) will denote either $\overset{\mathtt{a}}{\sim}$ and $\overset{\mathtt{n}}{\to}$ or the union of the `a`, `p` and `s` conversions and the `npz`-reduction respectively. The union of $\overset{\mathtt{cut}}{\longrightarrow}$ and $\overset{\mathtt{can}}{\longrightarrow}$ is denoted simply by $\to$, as it is *the* reduction of the system. In what follows we partition the reduction $\to$ in *erasing* and *non erasing*, giving the main ingredient for the statement of the conservation theorem. For some of the rules the distinction is clear: weakening on box and reductions to 0 are clearly erasing whole parts of the net, possibly containing the source of an infinite reduction. For others the decision is less clear (like for canonical reductions, or the harmless ones involving (co)weakenings) and has been guided by more technical reason, usually to make some proofs easier.

Definition 2. DiLL's **non-erasing** reduction $\overset{\neg\epsilon}{\to}$ is the context closure of the following cut-elimination steps (see Figure 4):

$$\neg\epsilon := \otimes/\parr + 1/\bot + !/? + !_2/? + !\pi/? + \dagger/!\pi + \dagger/?_2$$

with $\pi \neq 0$ and $\dagger$ denoting either a box (possibly $!0$), a codereliction or a cocontraction.

DiLL's **erasing** reduction $\xrightarrow{\epsilon}$ is the context closure of the other cut-elimination steps and the canonical reduction ones (see Figures 4, 5):

$$\epsilon := !0/? + \dagger/!0 + \dagger/?_0 + !_0/\varepsilon + \mathfrak{n} + \mathfrak{p} + \mathfrak{z}$$

with $\dagger$ denoting either a box (possibly $!0$), $!_0$, $!$, or $!_2$, and ε denoting either a box, $?_0$, $?$ or $?_2$.

Notice that $\xrightarrow{\neg\epsilon}$ and $\xrightarrow{\epsilon}$ are disjoint and their union yields the whole set of reduction rules introduced so far for DiLL. Moreover, non-erasing reduction is a subset of cut-elimination.

2.3. *Examples and Remarks*

Let us get to know DiLL by discussing some examples. Consider the simple net π_1 which is at the top left corner of Figure 6. The depth of π_1 is 1, it has three cuts at depth 0 (a clash, typed by $\mho$, a deadlock and a reducible cut of kind $!/!\rho$) and one cut at depth 1, of kind $?/!_2$. We have explicitly labelled only the wires with type $\mho$, as the other types are irrelevant to the example. This policy will be taken almost everywhere in the paper: types are left implicit whenever irrelevant or deducible. We will now describe each of the depicted reduction steps.

$\pi_1 \to \pi_2$. The first step fires on the cut inside the box. It is non-deterministic, creating a sum of two addends, depending on which premise of the cocontraction gets the dereliction (the other premise receiving a weakening instead). Here is a major difference with Linear Logic, in which nets have an essentially deterministic reduction[7]: DiLL's cut-elimination can *create* sums from simple nets. Notice that the sum duplicates the context at its same exponential depth but it does not spread outside the box: this is akin to the additive duplication as modeled by sliced proof nets of LL [Gir87a, PTdF10]. In this example however we will depict the polynets by gathering the context shared by their addends, so that π_2 is equivalently written as in the top right corner of Figure 6.

$\pi_2 \to \pi_3$. This step fires on the codereliction against box cut, which is a very delicate reduction step of DiLL. Indeed this kind of step affects the cuts on all the ports of the box. In particular in this example the deadlock has been changed into a reducible cut of kind $?_2/!_2$, and the clash has changed its kind even if staying a clash. This is another strong difference with respect to Linear Logic: in LL only the cuts on the principal port of a box can affect the cuts on the other ports, while in this example the cut is on an auxiliary port. It is because of this very difference that we have had to introduce lax typing, necessary in DiLL to get conservation while LL does not need it. We will discuss in detail this point later, in Figure 8.

$\pi_3 \xrightarrow{*} \pi_4 \xrightarrow{*} \pi_5$. The simple net π_4 is obtained from π_3 performing a $?_0/!$ step, eliminating (i.e. reducing to 0) one of the two addends of π_3, and (in the other addend) performing

[7] Actually, some side non-determinism is given by the additive unit $\top$ and the commutative additive step in presence of additive boxes [TdF03].

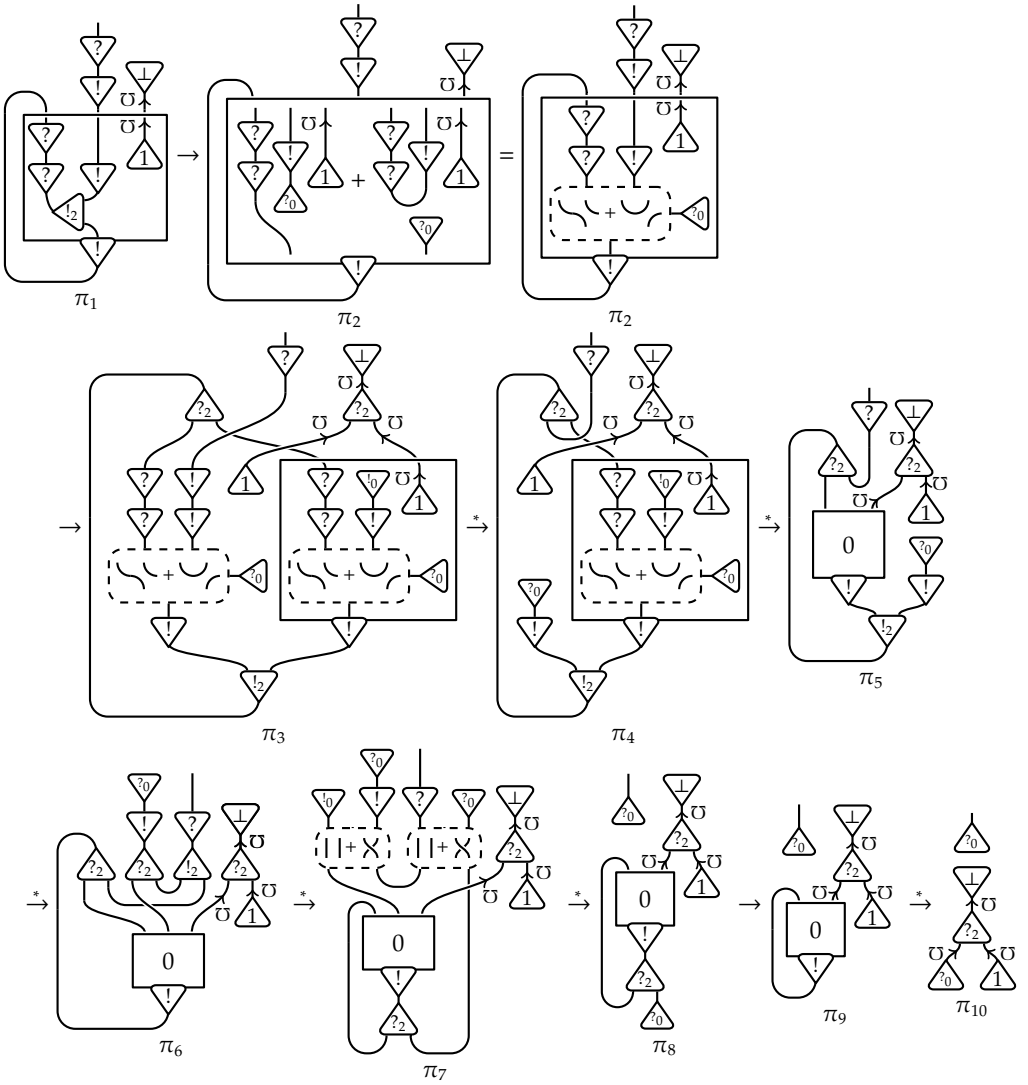


Figure 6. Example of a reduction in DiLL.

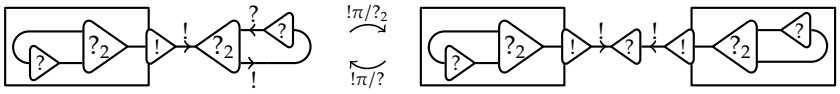


Figure 7. Example of laxly typed and switching acyclic nets without normal form. The types are written explicitly only for 0-depth wires.

two consecutive !/? steps. We then get π_5 by reducing the cuts in the box of π_4: such a reduction yields an empty box !0, since both the addends of the contents of the box of π_4 are erased. The attentive reader has certainly noticed that π_5 has been pictured by swapping the premises of its (co)contractions, benefiting from the commutative nature of these cells.

$\pi_5 \xrightarrow{*} \pi_6 \xrightarrow{*} \pi_7 \xrightarrow{*} \pi_8$**.** The reduction sequence $\pi_5 \xrightarrow{*} \pi_6$ fires on the $?_2/!_2$ cut and on the $!_2/!0$ one created in the first step. The $!_2/!0$ step shoves a copy of the cocontraction inside the empty box, making it disappear. The polynet π_7 is the result of the reduction of the two cuts in π_6 of kind $!/?_2$ and $?/!_2$ respectively. Since both the kinds are non-deterministic, π_7 is a sum of four addends. Notice that the number of steps needed to achieve π_7 from π_6 is not two but three, due to the context duplication generated by the first step. Finally, the simple net π_8 is obtained by reducing three addends of π_7 to 0, by steps of kind $!_0/?$, $!/?_0$ and $!/!0$ respectively (this last one resulting in 0 since it opens an empty box), and by performing an !/? and an $!_0/!0$ step on the remaining addend.

$\pi_8 \to \pi_9 \xrightarrow{*} \pi_{10}$**.** The net π_9 is the result of a canonical step $\mathbf{n}$. Notice that π_9 is a normal form for the reduction not considering the optional rules of Figure 5. Be aware that π_9 contains however two irreducible cuts (a deadlock and a clash): since nets are just laxly typed and might contain vicious cycles, being a normal form does not imply being cut-free. Besides, if we take into account the whole set of rules defined so far, π_9 can be further reduced by one $\mathbf{z}$ and then one $!_0/?_0$ step, achieving the normal form π_{10}. Notice that π_8 is not SN, as an infinite reduction starts from it (by reducing the $?_2/!0$ cut, then those created by that reduction and so on...) while π_9 and π_{10} are.

As in Linear Logic [PTdF10], infinite reductions can be produced in DiLL for two distinct reasons: because of the presence of a sort of *vicious* cycle (i.e. the switching cycles as defined in subsection 2.5) or because of the very liberal typing discipline. The infinite reductions starting from the net π_8 of Figure 6 (as well as those from the polynets $\pi_3, \dots, \pi_7$) are in fact examples of the first sort. Indeed, by replacing the one and bottom by a weakening and an empty box, respectively, we get a net which is simply typable (see section 5) but still has the same infinite reductions. As an example of infinite reduction due to lax typing we have the net in Figure 7, where the reduction makes a loop between two laxly typed simple nets that have no switching cycle. This paper deals mainly with the latter sort of infinite reductions, as our main results are restricted to switching acyclic nets. Besides switching acyclicity, the other basic notion we adopt in this paper is the one of lax typing. Let us discuss it with the example in Figure 8. The net ξ at the top left corner of the figure has two reducible cuts resp. of kind $?/!\rho$ and $!/!\rho$ and one clash, which

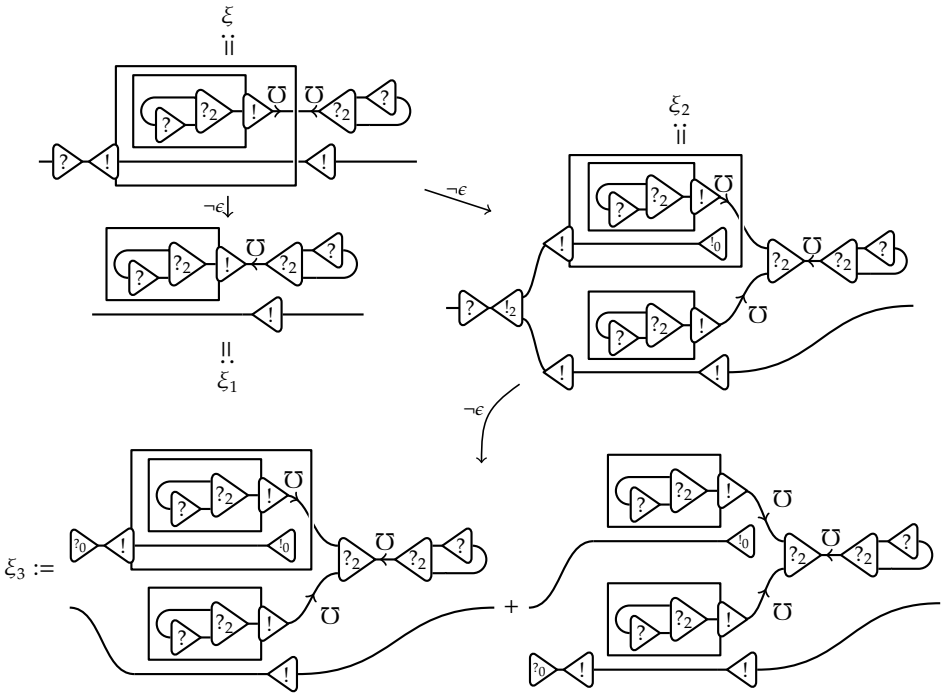


Figure 8. Example of the importance of the lax typing in DiLL. Only the ℧ type is written explicitly.

is a contraction against an auxiliary port of the box. By reducing the codereliction against box cut and then all the non-erasing cuts created at each step, we achieve the net ξ_3 at the bottom of the figure, which is a $\neg\epsilon$-normal form. This means ξ is $\mathsf{WN}_{\neg\epsilon}$. On the other hand, the reduction of the dereliction against box cut in ξ results in the net ξ_1, which is a normal form also, since the contraction against box cut has ℧ type. Notice that this ℧ cut is the residue of a clash in ξ that has became a priori reducible in ξ_1, but lax typing prevents us from reducing it. Indeed, if we could reduce this cut in ξ_1 we would obtain the infinite reduction of Figure 7, getting therefore a counterexample to conservation, as $\xi \in \mathsf{WN}_{\neg\epsilon}$. Lax typing stands here precisely to forbid such a counterexample.

Here is another difference with respect to LL, where conservation can be proved (at least for the sliced nets) without introducing any sort of typing whatsoever [PTdF10]. In a completely untyped setting, both LL and DiLL have clashes becoming reducible after some non-erasing steps. In LL though this fact is independent from which reduction is performed, which is not the case in DiLL: in Figure 8, for example, the clash in ξ becomes reducible in ξ_1 but remains irreducible in ξ_2 and in all of its reducts.

As already written in the Introduction, the main idea behind our proof of conservation is to follow [PTdF10], inferring Conservation through Gandy/Nederpelt's Lemma [Gan80, Ned73]. This lemma states that a rewriting system which is confluent and has an increasing measure enjoys WN = SN. The statement has been improved by Bezem and

Klop [Ter03] replacing confluence by local confluence. The goal would then be to achieve Conservation through local confluence of non-erasing reduction. However this plan fails when applied directly to DiLL. A first obstacle, which can however be overcome, is that local confluence already fails for general reduction in differential nets: one needs an equivalence $\sim$ (at least $\overset{\mathtt{a}}{\sim}$) and a canonical reduction (at least $\xrightarrow{\mathtt{n}}$) to achieve confluence modulo $\sim$ (local and non) of DiLL's reduction, as is shown in [Tra09a][8]. This problem is surmounted extending DiLL with the canonical conversions and reductions and upgrading Bezem and Klop's lemma in a rewriting modulo setting, as we do in Lemma 37. In particular this new setting needs to replace the local confluence condition with the local confluence modulo $\sim$ and local coherence with $\vdash$ (the symmetric relation generating $\sim$).

However the next obstacle we stumble upon is that the introduction of the equivalence and canonical rules fixes the problem for DiLL's reduction but not for its non-erasing restriction. An example is given by the same net in Figure 8: ξ $\neg\epsilon$-reduces to two distinct non-erasing normal forms. The critical pair is made of a dereliction against a box on one side and a coderelication against the same box on the other. In order to make it converge one needs to perform some evidently erasing steps on ξ_2's side. There seems to be a mismatch in the way dereliction and coderelication behave when cut against a box, as the latter leaves behind an unopened copy of it while the former does not. Such a copy will necessarily have to be erased to close the confluence diagrams.

For these reasons, we introduce $\mathsf{DiLL}_{\partial\varrho}$ in the next subsection, a new system essentially replacing the dereliction and coderelication with two cells, the linear query and the linear substitution respectively (Figure 9). The two new cells have a more symmetrical behavior when cut against a box, as *both* leave an unopened copy behind. $\mathsf{DiLL}_{\partial\varrho}$ enjoys non-erasing local confluence modulo $\sim$[9]. We will thus apply Lemma 37 to $\mathsf{DiLL}_{\partial\varrho}$, inferring Conservation for that system (section 3), and then we will transfer the result to DiLL (section 4).

2.4. *The System* $\mathsf{DiLL}_{\partial\varrho}$

In $\mathsf{DiLL}_{\partial\varrho}$ the roles of dereliction and coderelication are replaced by new cells, allowing non-erasing reduction to be locally confluent[10] modulo $\sim$ (Definition 3, Proposition 8). Having a locally confluent non-erasing reduction is an essential ingredient for proving Conservation Theorem via Lemma 37. However, as discussed above, this fails in DiLL, hence we pass via $\mathsf{DiLL}_{\partial\varrho}$. Actually $\mathsf{DiLL}_{\partial\varrho}$ has its own interest both for its computational behaviour (somewhat smoother than DiLL) and its tight link with a categorical interpretation of differential linear logic [Fio07]: ∂ and ϱ fill the roles of the annihilation and creation operators of [Fio07] respectively.

$\mathsf{DiLL}_{\partial\varrho}$ is obtained by removing from DiLL the dereliction and coderelication cells and

[8] Indeed [Tra09a] proves DiLL enjoys Church-Rosser modulo, which is a property stronger than confluence modulo.

[9] Indeed it also enjoys full non-erasing Church Rosser modulo, though we will not show it as it is beyond our objective.

[10] The reduction is indeed Church-Rosser modulo $\overset{\mathtt{c}}{\sim}$, see [Tra09b].

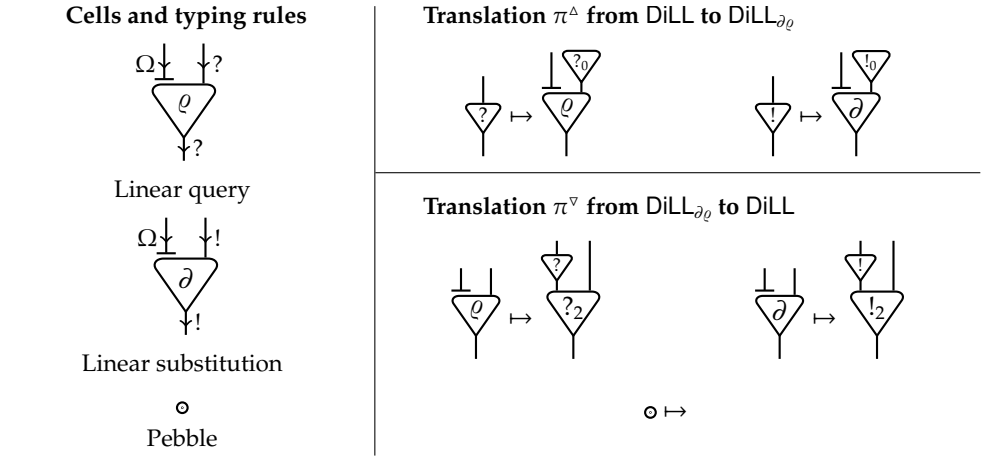


Figure 9. The cells ∂ and ϱ of DiLL$_{\partial\varrho}$, and the translation from and to DiLL. The cells are *not* commutative, but may be drawn with swapped inactive ports, hence the special marking of the left port.

adding in the two cells ∂ (**linear substitution**) and ϱ (**linear query**) shown together with their lax typing rules in Figure 9. As usual lax typing should be intended downward closed with respect to $\sqsubseteq$. We also add a cell, called **pebble**, with no ports (which is therefore a tolerated exception to each cell having a principal port), whose only role is to make the size always increasing under non-erasing reduction[11]. For the sake of clarity we define pebbles as an integral part of DiLL$_{\partial\varrho}$; they can however be completely disregarded if DiLL$_{\partial\varrho}$ is to be considered a stand-alone system. In the same figures translations from DiLL nets to DiLL$_{\partial\varrho}$ ones and back are defined, by giving constructs substituting the missing cells. We denote these translations by π^{Δ} and π^{∇} respectively.

All the reduction and conversion rules of DiLL (Figures 4 and 5) save for the ones involving tensor, one, dereliction or codereliction are still valid in DiLL$_{\partial\varrho}$. The new rules are listed in Figure 10. The main differences with respect to DiLL are in the definition of the steps dealing with ∂ and ϱ (easily deducible by the translation into DiLL), and in the addition of pebbles in those DiLL steps that, if we were in a simple type setting, would have decreased the type of cuts.

Analogously to DiLL, we split DiLL$_{\partial\varrho}$ reduction rules into erasing and non-erasing ones.

Definition 3. DiLL$_{\partial\varrho}$'s **non-erasing** reduction $\xrightarrow{\neg\epsilon}$ is the context closure of the following cut-elimination steps (see Figures 4 and 10):

$$\neg\epsilon := \otimes/\parr + 1/\bot + \partial/\varrho + !_2/\varrho + !\pi/\varrho + \dagger/!\pi + \dagger/?_2$$

with $\pi \neq 0$ and $\dagger$ denoting either a box (possibly $!0$), a ∂ or a cocontraction.

[11] Therefore the role of pebbles covers what is done by labeling terms with natural numbers in [Sør97].

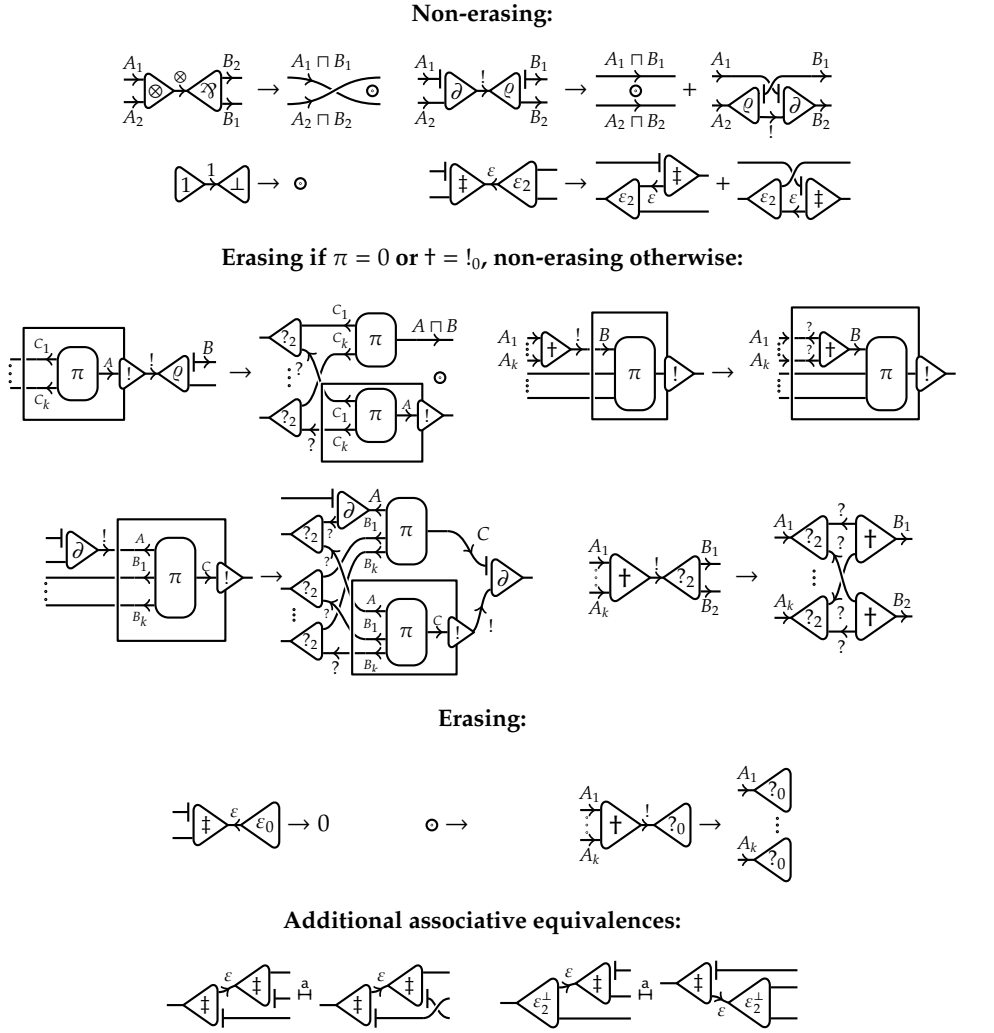


Figure 10. Reduction and new conversion rules of $\mathsf{DiLL}_{\partial\varrho}$. The symbols ε and ‡ can be either ! and ϱ or ? and ∂ respectively. The symbol † denotes either $!\xi$ (i.e. a box) $!_2$ or $!_0$

$\mathsf{DiLL}_{\partial\varrho}$'s **erasing** reduction $\xrightarrow{\epsilon}$ is the context closure of the other cut-elimination steps and the canonical reductions (see Figures 10 and 5):

$$\epsilon := !0/? + \dagger/!0 + \dagger/?_0 + !_0/\ddagger + \mathsf{pebble} + \mathsf{n} + \mathsf{p} + \mathsf{z}$$

with † denoting either a box (possibly !0), ∂, $!_0$ or $!_2$, and ‡ denoting either a box, ϱ, $?_0$ or $?_2$.

If we now take ξ^{Δ}, with ξ defined as in Figure 8, we can see that the generated critical peak can now be closed under non-erasing reduction. Indeed we will show the general case to hold in Figure 12 while proving Proposition 8.

2.5. *Switching Acyclicity*

Cut-elimination can be performed without any reference to any form of correctness. However we noticed in subsection 2.3 that certain cycles, morally corresponding to logical errors, give a deranged computational behaviour (even in linear logic and even for simply typable nets). Purely graph-theoretical conditions, called *correctness criteria*, have been presented in order to characterize the set of "correct" nets. We here give the most known of such correctness criteria, *switching acyclicity*, presented originally by Danos and Regnier [DR89] for the multiplicative fragment of linear logic, and then extended to DiLL in [ER06][12].

First of all, one sets a subset of the symbols of the system to be **switching**. In DiLL the switching symbols are just $\parr$ and $?_2$, in $\mathsf{DiLL}_{\partial\varrho}$ ϱ is added as switching too. All the cells in a net with a switching symbol are called switching too. One then defines **switching paths** as intuitively graph-theoretical paths on a net which cannot pass through two premises of the same switching cell. Formally a switching path ϕ of a simple net α is a non-repeating sequence of ports $p_1 \cdots p_k$ in $\mathsf{p}_0(\alpha)$ such that:

— two consecutive ports p_i and p_{i+1} are either in the same cell or in the same wire, and $k \geq 2$ (i.e. the sequence defines an actual path);
— at most two ports from each cell appear in ϕ (which forces the path to alternate cells and wires and prevents it from recrossing or ending on a cell already traversed by it);
— no two ports in ϕ are the premises of the same switching cell (the actual switching condition).

Notice in particular that boxes are non-switching cells, so paths may enter and exit any of their ports regardless of their contents (a sort of *black box* principle). A **switching cycle** is either a loop or a switching path whose starting and ending ports are on the same cell. Moreover switching paths are considered disjoint if they have not only no ports, but also no cells in common.

A simple net α is switching acyclic if it contains no switching cycles and for every box $!\rho$ with depth 0 in α, ρ is switching acyclic; a differential net π is **switching acyclic** (AC) if every simple net of π is switching acyclic.

What follows are results renowned in literature, namely that switching acyclicity is preserved by reduction. As this is usually proved by showing that reduction does not create switching paths (in particular cycles), and we will need such more general result later in subsection 4.3, we state such a result more precisely (Lemma 4 below), while skipping on the details of the proof of Proposition 5.

We say that a relation R defined on polynets with the same free ports does not create switching paths if whenever $\pi \; R \; \rho$, with $p, q \in \mathsf{fp}(\pi) = \mathsf{fp}(\rho)$ and with a switching path linking p and q in an addend of ρ, then there is also one linking the same ports in π.

Lemma 4. Reduction $\to$ and equivalence $\sim$ both do not create switching paths.

[12] To be precise, [ER06] introduces switching acyclicity in the promotion-free fragment of DiLL, however its generalization to boxes is trivial.

Proof. Let us define the following more refined property of a relation R, that implies that R does not create switching paths:

$$\text{if } \pi\ R\ \rho \text{ and } \phi_i \text{ is a family of disjoint switching paths in } \rho \text{ linking some of its free ports. Then there are disjoint paths } \phi'_i \text{ linking the same ports in } \pi. \tag{1}$$

First we see that (1) is preserved by context closure (which in fact is not the case in general for not creating switching paths). Indeed suppose $\omega[\pi]\ \tilde{R}\ \omega[\rho]$ with $\pi\ R\ \rho$. Then any switching path ϕ_i in $\omega[\rho]$, as it may traverse ρ multiple times, induces a possibly empty family ϕ_{ij} of k_i paths in ρ and a family ψ_{ij} of $k_i + 1$ paths in $\omega[\ ]$ so that gluing together $\psi_{i1}, \phi_{i1}, \psi_{i2}, \dots, \phi_{ik_i}, \psi_{ik_i+1}$ we get back ϕ_i in $\omega[\rho]$. Moreover given the disjoint family ϕ_i, all the ϕ_{ij} for any pair of distinct i and j are also disjoint. We thus get by definition of (1) disjoint paths ϕ'_{ij} in π, so that gluing back $\psi_{i1}, \phi'_{i1}, \dots, \phi'_{ik_i}, \psi_{ik_i+1}$ we get the needed family of disjoint switching paths in $\omega[\pi]$.

To prove the lemma one then needs just to check (1) for all the reduction and coversion rules, which is easily carried out by inspecting them. □

Proposition 5. Reduction $\to$ and equivalence $\sim$ both preserve switching acyclicity.

Notice that in $\mathsf{DiLL}_{\partial\varrho}$ the status of the new symbols (with ϱ switching and ∂ not) is in fact deducible from the corresponding translation into DiLL. It is indeed straightforward that π in $\mathsf{DiLL}_{\partial\varrho}$ is AC iff $\pi^{\triangledown}$ is, and the same for $\sigma^{\triangle}$.

3. The Conservation Theorem of $\mathsf{DiLL}_{\partial\varrho}$

In this section we prove that non-erasing reduction is perpetual in $\mathsf{DiLL}_{\partial\varrho}$, or equivalently, that $\mathsf{WN}_{\neg\epsilon}$ implies $\mathsf{SN}_{\sim}$ (Theorem 17). The proof splits this implication in two: subsection 3.1 shows that $\mathsf{WN}_{\neg\epsilon}$ implies $\mathsf{SN}_{\neg\epsilon\sim}$ and subsection 3.2 proves that $\mathsf{SN}_{\neg\epsilon\sim}$ implies $\mathsf{SN}_{\sim}$.

The fact that $\mathsf{WN}_{\neg\epsilon}$ implies $\mathsf{SN}_{\neg\epsilon\sim}$ (Lemma 10) is achieved by applying Lemma 37: we give a measure on nets (Definition 6) which is increasing under the $\neg\epsilon$ reduction and stable under conversion (Proposition 7), and we prove that $\xrightarrow{\neg\epsilon}$ is locally confluent modulo $\sim$ (Proposition 8) and locally coherent with $\mapsto$ (Proposition 9). Let us underline that Propositions 8 and 9 (and hence conservation) need the switching acyclic hypothesis.

The fact that $\mathsf{SN}_{\neg\epsilon\sim}$ implies $\mathsf{SN}_{\sim}$ is obtained by showing that non-erasing steps can always be anticipated with respect to the erasing ones (Lemma 16) and that erasing reduction is $\mathsf{SN}_{\sim}$ (Proposition 12).

Throughout all this section we take as system $\mathsf{DiLL}_{\partial\varrho}$, so we may at times omit to mention it.

3.1. *From* WN *Non Erasing to* $\mathsf{SN}_{\sim}$ *Non Erasing*

The next definition uses the following notation from section 2: $\sigma(c)$ denotes the symbol of the cell c, and in case c is a box, $\sigma(c)$ is a net (its contents), so that $\#\sigma(c)$ is the number of addends of $\sigma(c)$, finally $\mathsf{c}_!(\pi)$ is the set of cells at any depth of π.

Definition 6. The *weight* $\mathrm{wg}(c)$ of a cell c at depth d in a net π is defined as follows

$$\mathrm{wg}(c) := \begin{cases} 1 & \text{if } c \text{ is a pebble,} \\ 1+d & \text{if } c \text{ is a cocontraction, coweakening, linear substitution or an empty box,} \\ (1+d)(2\#\sigma(c)-1) & \text{if } c \text{ is a non-empty box,} \\ 0 & \text{otherwise.} \end{cases}$$

The weight $\mathrm{wg}(\pi)$ of the net π is the sum of the weights of its cells: $\mathrm{wg}(\pi) = \sum_{c \in \mathsf{C}_!\,\pi} \mathrm{wg}(c)$.

Proposition 7. In $\mathsf{DiLL}_{\partial\varrho}$, if $\pi \xrightarrow{\neg\epsilon} \pi'$ then $\mathrm{wg}(\pi) < \mathrm{wg}(\pi')$, and if $\pi \sim \pi'$ then $\mathrm{wg}(\pi) = \mathrm{wg}(\pi')$.

Proof. First, one checks the property case by case for the non-erasing rules of Figure 9, of which we sketch some in the following. Consider contraction against a $\dagger$ cell c, for $\dagger = !_2$ or $!\pi$: the weight of the redex is the weight of c, which is always different from 0; the one of the contractum is twice the weight of c. Also, consider the reduction rule of linear substitution against box: the weight increases (independently from the contents of the box) since the contractum has at least one linear substitution more than the redex, as the contents of the box is not 0. As for conversion, consider the splitting of a sum in a box: the weight of the left-hand side net is $\mathrm{wg}(\pi) + \mathrm{wg}(\xi) + (2\#(\pi+\xi)-1)$, which is equal to $\mathrm{wg}(\pi) + \mathrm{wg}(\xi) + 1 + (2\#\pi - 1) + (2\#\xi - 1)$, which is the weight of the net on the right-hand side. Notice again the importance of the condition $\pi, \xi \neq 0$ to yield the previous equality.

One generalizes then to contexts noticing that $\neg\epsilon$ never decreases (resp. $\mapsto$ never changes) the number of addends in a sum, so $\mathrm{wg}(\omega[\alpha]) > \mathrm{wg}(\omega[\pi_\alpha])$ (resp. $\mathrm{wg}(\omega[\alpha]) = \mathrm{wg}(\omega[\pi_\alpha])$) whenever α, π_α is a $\neg\epsilon$ (resp. $\mapsto$) pair. □

Proposition 8. On the switching acyclic nets of $\mathsf{DiLL}_{\partial\varrho}$, the reduction $\xrightarrow{\neg\epsilon}$ is locally confluent modulo $\sim$, i.e. $\xleftarrow{\neg\epsilon}\xrightarrow{\neg\epsilon} \subseteq \xrightarrow{\neg\epsilon*}\sim\xleftarrow{\neg\epsilon*}$.

Proof. There are not many more critical peaks to check than in [Tra09b, Tra09a]. Apart from the confluence diagrams in common with DiLL (which can be seen to be 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.

For the former, Figure 11 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.

For the latter, Figure 12 shows the confluence diagram. □

Proposition 9. On switching acyclic nets of $\mathsf{DiLL}_{\partial\varrho}$, the reduction $\xrightarrow{\neg\epsilon}$ is locally coherent with $\mapsto$, i.e. $\mapsto\xrightarrow{\neg\epsilon} \subseteq \xrightarrow{\neg\epsilon*}\sim\xleftarrow{\neg\epsilon*}$.

Proof. As usual, one must check all coherence diagrams. The majority is trivial, or a variant over the ones already known for DiLL (see [Tra09a]). We show an interesting

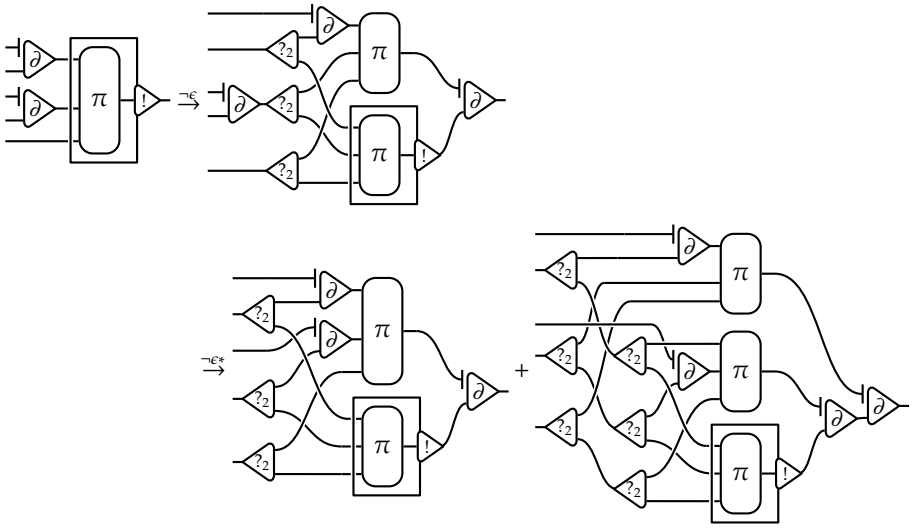


Figure 11. Reduction of the critical peak with two ∂ cells on a box. The box has only one other auxiliary port for brevity. Starting with the other ∂ cell is completely symmetric, and an a-conversion equates the two results.

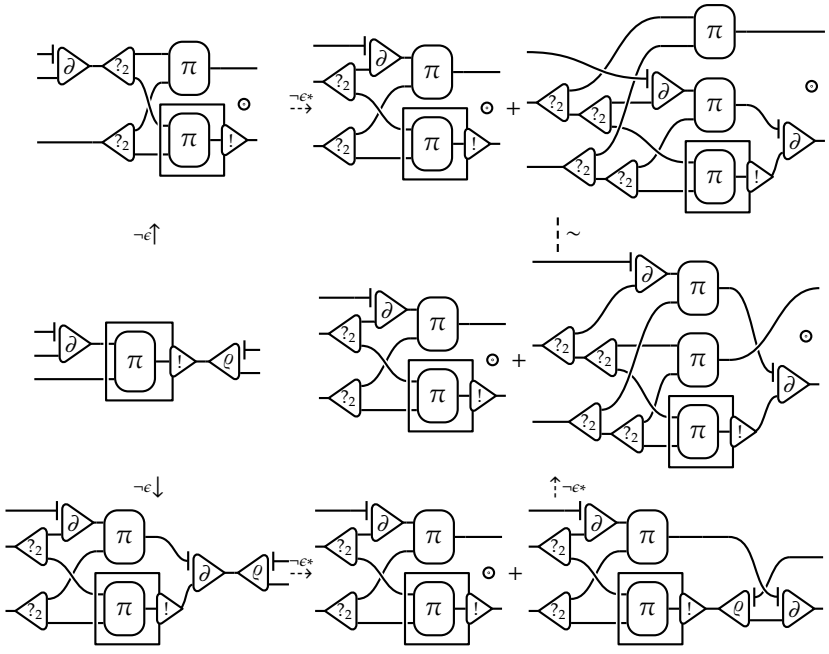


Figure 12. Confluence diagram of the ∂ on box on ϱ critical peak.

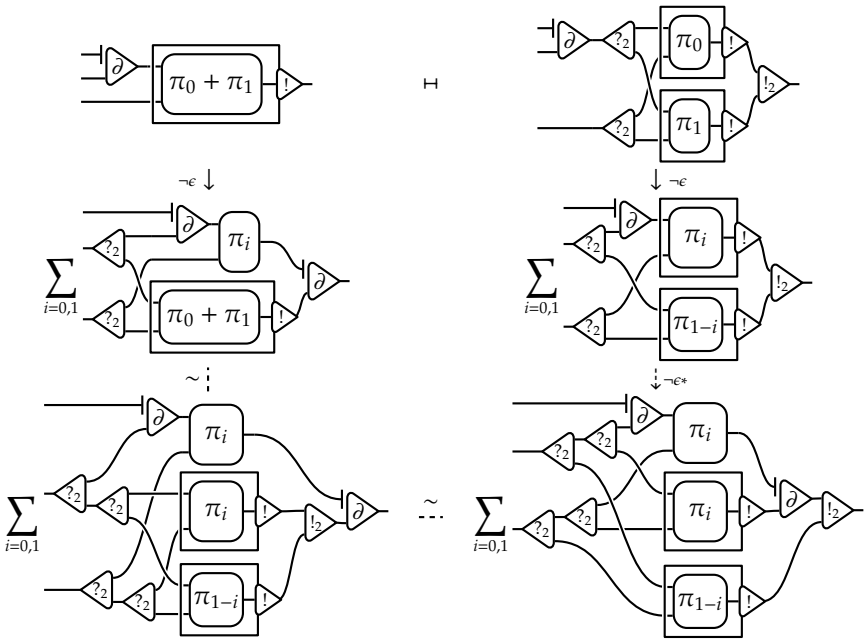
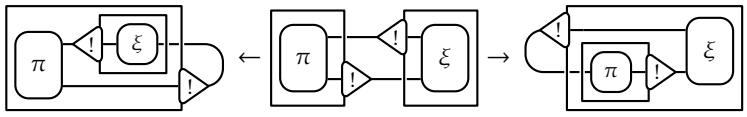
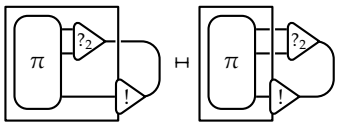


Figure 13. Coherence diagram of a ∂ cell on an s-conversion.



(a) counterexample to local confluence



(b) counterexample to local coherence

Figure 14. Counterexamples to local confluence and coherence of $\neg\epsilon$ with $\sim$ in presence of switching cycles. The latter is also a counterexample to the conservation theorem.

one in Figure 13, a ∂ cell on an s-convertible box. This is the only diagram that needs the "mixed" associativity rule between cocontraction and linear substitution. Recall that $\pi_0, \pi_1 \neq 0$ is required by the s-conversion, so that no step is forbidden by the non-erasing discipline. □

Propositions 8 and 9 need the switching acyclicity hypothesis: Figure 14(a) gives a counterexample to local confluence of $\xrightarrow{\neg\epsilon}$ (as well as of $\xrightarrow[\sim]{\mathsf{cut}}$) in presence of switching cycles;

analogously Figure 14(b) gives a counterexample of local coherence of $\xrightarrow{\neg\epsilon}$ with $\vdash$. In fact the latter is also a counterexample to the conservation property in presence of switching cycles.

Lemma 10 (conservation for non-erasing reduction). Given a net π in $\mathsf{DiLL}_{\partial\varrho}$, if π is switching acyclic and $\pi \in \mathsf{WN}_{\neg\epsilon}$ then $\pi \in \mathsf{SN}_{\neg\epsilon\sim}$.

Proof. Immediate consequence of Lemma 37 and the Propositions 7, 8 and 9. □

3.2. *From* $\mathsf{SN}_\sim$ *Non Erasing to* $\mathsf{SN}_\sim$ *in* $\mathsf{DiLL}_{\partial\varrho}$

We prove that non-erasing steps can always be anticipated with respect to the erasing ones (Lemma 16) and that erasing reduction is $\mathsf{SN}_\sim$ (Proposition 12). So we achieve conservation for $\mathsf{DiLL}_{\partial\varrho}$ (Theorem 17).

We recall again that, given a cell c, $\sigma(c)$ (resp. $\mathsf{deg}(c)$) denotes its symbol (resp. the number of its ports); namely if c is a box, $\#\sigma(c)$ is the number of the addends of the contents $\sigma(c)$ of c; finally, $\mathsf{c}_0(\pi)$ is the set of cells at depth 0 in π.

Definition 11. We define the **erasing weight** of a cell c and that of a net π by recursion on the depth of the net:

$$\mathsf{e}(c) := \begin{cases} 2 & \text{if } \sigma(c) \in \{!_2, ?_2, \partial, \varrho\}, \\ 1 & \text{if } \sigma(c) = !_0 \text{ or } c \text{ is a pebble}, \\ 2\,\mathsf{deg}(c)\big(2\,\#(\sigma(c)) - 1\big) + 2\,\mathsf{e}(\sigma(c)) & \text{if } c \text{ is a non-empty box}, \\ \mathsf{deg}(c) & \text{if } c \text{ is an empty box}, \\ 0 & \text{otherwise;} \end{cases}$$

$$\mathsf{e}(\pi) := \sum_{c \in \mathsf{c}_0(\pi)} \mathsf{e}(c).$$

Proposition 12. In $\mathsf{DiLL}_{\partial\varrho}$, if $\pi \xrightarrow{\epsilon} \pi'$ then $\mathsf{e}(\pi) > \mathsf{e}(\pi')$, and if $\pi \sim \pi'$ then $\mathsf{e}(\pi) = \mathsf{e}(\pi')$. In particular erasing reduction is $\mathsf{SN}_\sim$ in $\mathsf{DiLL}_{\partial\varrho}$.

Proof. As usual, one first checks the property on the erasing and conversion rules. We treat in detail only the more interesting cases.

For total pull reduction the weight clearly decreases since the box loses one auxiliary port. For cocontraction against an empty box, the latter having n ports, the weight of the redex is $2 + n$ while that of the contractum is $n + 1$. Notice in this step it is crucial that cocontractions weight 2 and not 1. For weakening against linear substitutions (or coweakening against linear query), the contractum weights 0 being the empty sum, while the redex weights 2.

As for the push conversion, using the notation of Figure 5, the right-hand side simple net weights $2n(2\,\#(\pi) - 1) + 2(2\,\#(\pi) + \mathsf{e}(\pi))$, with n the number of ports of the box, while the left-hand side weights $2(n+1)(2\,\#(\pi) - 1) + 2 + 2\,\mathsf{e}(\pi) = 2n(2\,\#(\pi) - 1) + 4\,\#(\pi) + 2\,\mathsf{e}(\pi)$, which is the same. Consider the sum conversion: at the left-hand side we have $2n(2\,\#(\pi) +$

$2\#(\xi) - 1) + 2(\mathsf{e}(\pi) + \mathsf{e}(\xi))$, with n the number of ports of the box, and at the right one $2n(2\#(\pi) - 1) + 2\,\mathsf{e}(\pi) + 2n(2\#(\xi) - 1) + 2\,\mathsf{e}(\xi) + 2n = 2n(2\#(\pi) + 2\#(\xi) - 1) + 2(\mathsf{e}(\pi) + \mathsf{e}(\xi))$.

One generalizes to contexts arguing symmetrically with respect to the proof of Proposition 12: since ϵ-reduction never increases (resp. $\vdash\!\dashv$ never changes) the number of addends in a sum, we have $\mathsf{e}(\omega[\alpha]) < \mathsf{e}(\omega[\pi_\alpha])$ (resp. $\mathsf{e}(\omega[\alpha]) = \mathsf{e}(\omega[\pi_\alpha])$) whenever α, π_α is an ϵ (resp. $\vdash\!\dashv$) pair. □

Lemma 13. We have $\xrightarrow{\epsilon}\!\!\vdash\!\dashv \subseteq \overset{\epsilon*}{\rightsquigarrow}\xrightarrow{\epsilon}$. Moreover, the equation is valid also if forbidding partial $\mathtt{p}$-reductions.

Proof. By direct inspection of the rules. Most of the ϵ-reductions cannot create nor modify a conversion instance, so the two can always be swapped. The interesting cases arise when:

— a neutral redex "blocks" an instance of a conversion $\overset{\mathtt{x}}{\vdash\!\dashv}$ for $\mathtt{x} \in \{\mathtt{a}, \mathtt{p}, \mathtt{s}\}$: it is in this case that an additional $\overset{\mathtt{a}}{\vdash\!\dashv}$ conversion is needed for the additional (co)contraction; $\overset{\mathtt{a}}{\vdash\!\dashv}$ can be performed since lax typing assures that the additional (co)contraction has the same !/? type as the neutral redex; we thus transform $\xrightarrow{\mathtt{n}}\!\!\vdash\!\dashv$ into $\vdash\!\dashv\!\!\vdash\!\dashv\xrightarrow{\mathtt{n}}$;

— a pull redex is then object of an $\mathtt{s}$-conversion: the diagram is closed by at most two $\mathtt{s}$-conversion (one coming from the original pull redex, in case it is partial, the other one from the original $\mathtt{s}$-conversion) followed by at most two total $\mathtt{p}$-reductions and at most two $\mathtt{n}$ ones. In any case the new $\mathtt{p}$-reductions can be total. □

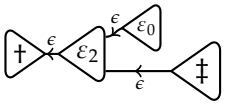
Lemma 14. We have $\overset{\epsilon*}{\rightsquigarrow} \subseteq {\sim}\xrightarrow{\epsilon*}$. Moreover, the equation holds when forbidding partial $\mathtt{p}$-reductions.

Proof. Using Lemma 13 and Lemma 39 on ϵ (possibly without partial $\mathtt{p}$- reductions) and with measure $\mathsf{e}(\pi)$ (Proposition 12) we get the result. □

Lemma 15. If we forbid partial $\mathtt{p}$-reductions we have $\xrightarrow{\epsilon}\!\!\xrightarrow{\neg\epsilon} \subseteq \xrightarrow{\neg\epsilon}\!\!\xrightarrow{*}$.

Proof. For all erasing reductions but the $\mathtt{n}$ one, the result is achieved by noticing that the erasing step (by hypothesis different from partial $\mathtt{p}$) 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 are then easily handled. Partial pull reductions need conversions to swap with non-erasing ones, so we explicitly left it out.

The $\mathtt{n}$ 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 $\mathtt{n}$-reduction. The situation is the following



where ε is either ? or ! and † can interact both with the (co)contraction and with ‡. Then

— if † is duplicated by ε_2 (i.e. † is a box, cocontraction or contraction): one copy can then reduce by a non-erasing step with ‡, the other will be erased by ε_0 and $\mathfrak{n}$-reductions will close the diagram;
— if † is ∂ or ϱ, the reduction against ε_2 generates a sum: in one addend we can follow with the non-erasing † on ‡, then close by erasing the other addend by ε_0 on ∂ or ϱ and the weakening-coweakening pair in the addend left;
— if † is a box and ε_2 a cocontraction, then we can make ε_2 enter † first, then fire the † on ‡ redex, and then make all the residuals of ε_0 $\mathfrak{n}$-reduce with all the residuals of ε_2, possibly making them enter the residuals of †. □

Lemma 16. In $\mathsf{DiLL}_{\partial\varrho}$ we have $\overset{\epsilon*}{\rightsquigarrow}\overset{\neg\epsilon}{\rightsquigarrow} \subseteq \overset{\neg\epsilon}{\rightsquigarrow}\overset{*}{\rightsquigarrow}$.

Proof. Take $\pi \overset{\epsilon*}{\rightsquigarrow}\overset{\neg\epsilon}{\rightsquigarrow} \xi$. First, we purge the chain of any partial $\mathfrak{p}$-reduction by substituting them with an $\mathfrak{s}$-conversion followed by a total $\mathfrak{p}$-reduction and some $\mathfrak{n}$-reductions. Using Lemma 14 we get $\pi \sim\overset{\epsilon*}{\rightarrow}\overset{\neg\epsilon}{\rightarrow}\sim \xi$, again with no partial $\mathfrak{p}$ reduction. Iterating Lemma 15, we can shove at least a $\neg\epsilon$-reduction at the start and get the result. □

Theorem 17 (conservation for $\mathsf{DiLL}_{\partial\varrho}$). Given a net π in $\mathsf{DiLL}_{\partial\varrho}$, if π is switching acyclic and $\pi \in \mathsf{WN}_{\neg\epsilon}$, then $\pi \in \mathsf{SN}_{\sim}$.

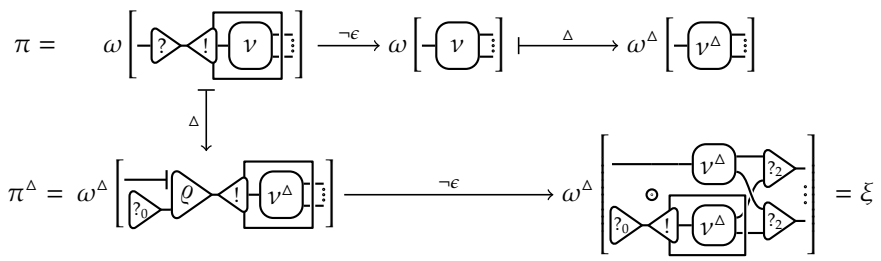
Proof. Let $\pi \in \mathsf{WN}_{\neg\epsilon}$, which by Lemma 10 means that $\pi \in \mathsf{SN}_{\sim}$ for non-erasing reduction. Suppose we have an infinite chain R of reductions interleaved by $\sim$-conversions starting from π. If we are able to build by induction an infinite chain of non-erasing 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 $\pi\ (\overset{\neg\epsilon}{\rightsquigarrow})^k\ \pi_k$, where the exponent k is the number of $\overset{\neg\epsilon}{\rightsquigarrow}$ steps of R_k ($\pi_0 := \pi$), and an infinite reduction R' starting from π_k. By Proposition 12 R' cannot be solely made of erasing steps, so it must necessarily start with a segment of the shape $\pi_k \overset{\epsilon*}{\rightsquigarrow}\overset{\neg\epsilon}{\rightsquigarrow} \pi'$, with an infinite reduction starting from π'. Now Lemma 16 ensures that we have $\pi\ (\overset{\neg\epsilon}{\rightsquigarrow})^{k+1}\ \pi_{k+1}$ with $\pi_{k+1} \overset{*}{\rightsquigarrow} \pi'$, so that π_{k+1} also has an infinite reduction. □

4. Transferring Conservation from $\mathsf{DiLL}_{\partial\varrho}$ to DiLL

In this section we achieve the conservation theorem for DiLL. Let π be a switching acyclic DiLL net, we will prove that $\pi \in \mathsf{WN}_{\neg\epsilon}$ implies $\pi \in \mathsf{SN}_{\sim}$. The idea is to pass through $\mathsf{DiLL}_{\partial\varrho}$ using the translation π^{Δ} given in Figure 9: by the conservation of $\mathsf{DiLL}_{\partial\varrho}$ we have $\pi^{\Delta} \in \mathsf{WN}_{\neg\epsilon} \implies \pi^{\Delta} \in \mathsf{SN}_{\sim}$, so what remains to prove is $\pi \in \mathsf{WN}_{\neg\epsilon} \implies \pi^{\Delta} \in \mathsf{WN}_{\neg\epsilon}$ and $\pi^{\Delta} \in \mathsf{SN}_{\sim} \implies \pi \in \mathsf{SN}_{\sim}$. The latter is an immediate consequence of $\mathsf{DiLL}_{\partial\varrho}$ simulating DiLL $\twoheadrightarrow$ (Lemma 18). On the contrary, $\pi \in \mathsf{WN}_{\neg\epsilon} \implies \pi^{\Delta} \in \mathsf{WN}_{\neg\epsilon}$ is quite tough and the rest of this section is devoted to achieve it (Lemma 29). The problem is that both DiLL and $\mathsf{DiLL}_{\partial\varrho}$ needs erasing steps to simulate the reduction of the other one, hence a non-erasing normalizing chain for π cannot be translated directly to a non-erasing

normalization of π^Δ. The core of the problem is DiLL's dereliction on box rule:



By induction hypothesis we can suppose that the translation $\omega^\Delta[\nu^\Delta]$ of the contractum of π is $\mathsf{WN}_{\neg\epsilon}$, but what about π^Δ and its contractum ξ? If we are to use a normalizing chain of $\omega^\Delta[\nu^\Delta]$ to produce one for ξ we have to do "more work", as we must account for a boxed copy of ν^Δ and all the reductions it might spawn. We will tackle this in Lemma 28 by defining a relation that holds for $\omega^\Delta[\nu^\Delta]$ and ξ but is also sufficiently general to be preserved under non-erasing reduction and sufficiently precise to allow transferring a normalizing chain of reductions. This relation is the composition $\twoheadrightarrow\rightsquigarrow$ of a relation $\twoheadrightarrow$ called *unfolding* (subsection 4.3) and a rewriting $\rightsquigarrow$ (subsection 4.2). Yet another remark: to make Lemma 28 work, we require the context $\omega[\,]$ to be linear, as otherwise the above mentioned relation gets too complicated to handle. Nevertheless this is enough to get $\pi \in \mathsf{WN}_{\neg\epsilon} \Longrightarrow \pi^\Delta \in \mathsf{WN}_{\neg\epsilon}$ thanks to a standardization property that has its own interest (Theorem 21, shown in subsection 4.1).

Lemma 18 ($\mathsf{DiLL}_{\partial\varrho}$ simulates DiLL). If in DiLL $\pi \twoheadrightarrow \xi$ then $\pi^\Delta \overset{+}{\twoheadrightarrow} \xi^\Delta$ in $\mathsf{DiLL}_{\partial\varrho}$. In particular, if π^Δ is $\mathsf{SN}_\sim$, then so is π in DiLL.

Proof. The proof is an exercise in reducing nets. For equivalence relations, it is immediate that $\pi \mapsto \xi$ implies $\pi^\Delta \mapsto \xi^\Delta$. For non-erasing reductions we refer to the figures in the proof of Lemma 29, where for each non-erasing redex $\lambda \to \mu$ involving a dereliction or codereliction in DiLL, both μ^Δ and the first step in reducing λ^Δ is shown. It is straightforward to see that reducing further all such immediate contracta with erasing steps gives μ^Δ in all the cases. As for erasing steps, the only relevant cases are the reduction to 0 with (co)weakening on (co)dereliction, for which the result is immediate. □

4.1. *The Standardization Theorem*

What we will prove in the following is the equivalent of the standardization theorem in λ-calculus [Bar81, Theorem 11.4.7]. As the standardization we need in Lemma 29 is for non-erasing reduction in DiLL, we state the theorem to encompass also this case. We recall that the depth of a step $\omega[\pi]\ \tilde{R}\ \omega[\xi]$ with $\tilde{R}$ the context closure of R and $\pi\ R\ \xi$ is the number of nested boxes containing the hole in $\omega[\,]$. Let us denote by $\to_0$ (resp. $\to_{>0}$) a reduction happening at depth 0 (resp. greater than 0), and similarly for $\mapsto$, $\sim$ and the combination $\twoheadrightarrow$. For the sake of clarity we will first prove a postponement property for non-erasing reduction, then we generalize to full $\twoheadrightarrow$, and finally we arrive

to standardization of both. Notice that all results in this subsection do not need the switching acyclicity hypothesis.

Lemma 19. $\overset{\neg\epsilon}{\to}_{>0}$ can be postponed with respect to $\overset{\neg\epsilon}{\to}_0$, i.e. $\overset{\neg\epsilon^*}{\to} \subseteq \overset{\neg\epsilon^*}{\to}_0 \overset{\neg\epsilon^*}{\to}_{>0}$.

Proof. We aim at applying Lemma 39, taking $\overset{1}{\to}$ and $\overset{2}{\to}$ to be the expansions $\overset{\neg\epsilon}{\leftarrow}_0$ and $\overset{\neg\epsilon}{\leftarrow}_{>0}$ respectively. We thus need an increasing measure $|\pi|$, and to obtain it we tweak the system: we add pebbles as in $\mathsf{DiLL}_{\partial\varrho}$, we redefine all non-erasing reductions to throw in a pebble in each addend of the contractum and finally we take $|\pi|$ to be the number of pebbles in π at any depth. Clearly postponement in this system entails the same in DiLL, as the latter can be recovered by just forgetting pebbles. The measure increases on all steps as no erasing is done. It remains to show that

$$\overset{\neg\epsilon}{\to}_{>0}\overset{\neg\epsilon}{\to}_0 \subseteq \overset{\neg\epsilon}{\to}_0\overset{\neg\epsilon^*}{\to},$$

considering also pebbles. In all cases, the deeper step cannot have created the 0-depth redex, so the latter step can always be performed first. Then the deeper one can be performed on all the residuals of its redex, whether they have been duplicated additively, or by a reduction on the box containing it which may have also lowered its depth. □

Lemma 20. Take $\sim$ and $\overset{\mathsf{can}}{\to}$ to be $\overset{\mathtt{a}}{\sim}$ and $\overset{\mathtt{n}}{\to}$ respectively. Then $\to_{>0}\cup\mapsto_{>0}$ can be postponed with respect to $\rightrightarrows_0$, i.e. $\overset{*}{\rightrightarrows} = (\rightrightarrows_0 \cup \to_{>0} \cup \mapsto_{>0})^* \subseteq \overset{*}{\rightrightarrows}_0 (\to_{>0} \cup \mapsto_{>0})^* = \overset{*}{\rightrightarrows}_0\overset{*}{\sim}_{>0}$.

Proof. We adapt the above proof to this case. However we need to tweak the system further to avoid erasing steps from deleting the contribution of pebbles to the measure. Let us therefore introduce special nets $\underline{n}$ for $n \in \mathbb{N}$, considered to have any number and type of free ports. We redefine $|\pi|$ to be the total number of pebbles plus the sum of n for any $\underline{n}$ in π at any depth. We define how $\underline{n}$ interacts with other nets by setting $\omega[\underline{n}] = \underline{n + |\omega|}$ for $\omega[\ ]$ linear, i.e. it erases the rest of the context just like 0, "saving" however its measure. To prevent 0-boxes from erasing part of the measure, we consider them as being $!\underline{0}$.

Now we redefine all reductions but $?_0/!\pi$, $?_0/!$ and $!_0/?$ to introduce a pebble as above. Of the remaining rules, we redefine the first adding $|\pi|+1$ pebbles in the contractum; the latter two, we change them to yield $\underline{1}$. Finally, in order to close the needed diagram, we add two dummy reductions: one creating a pebble from nothing and the other taking $\underline{n}$ to $\underline{n+1}$. The measure $|\pi|$ now increases strictly on all reduction steps (and is invariant under $\overset{\mathtt{a}}{\sim}$). The delicate cases are the truly erasing ones: opening a box containing only $\sum \underline{n_i}$ (including $!0$ which has become $!\underline{0}$) is increasing due to $\underline{n}$ saving the measure of the net to which it is glued (and there is also a new pebble in it here); a box entering a $!(\sum \underline{n_i})$ is settled for the same reason, as the new contents of the box are indeed a linear context given by the entering box and applied to the old contents; $?_0/!\pi$, $?_0/!$ and $!_0/?$ again restore the erased part of the measure. Moreover, normal DiLL can be promptly recovered by forgetting both pebbles and $\underline{n}$ nets, and the redefined reductions map to normal ones, possibly loosing the new steps. In particular proving postponement in the tweaked system entails the same property in DiLL.

Now in order to use Lemma 39, we show that

$$(\rightarrow_{>0} \cup \mapsto_{>0})\twoheadrightarrow_0 \subseteq \twoheadrightarrow_0 \overset{*}{\twoheadrightarrow}.$$

The reasoning goes as in the above lemma, as a-conversions do not pose any serious problem. The only really different case is when the 0-depth step erases the location of the deeper one. In this case making the 0-depth one first gives necessarily a strictly lower number of pebbles or $\underline{n}$-net, so that dummy steps are needed to close the diagram again. □

Notice that postponement of $\twoheadrightarrow_{>0}$ with respect to $\twoheadrightarrow_0$ does not give that $\overset{*}{\twoheadrightarrow} \subseteq \overset{*}{\twoheadrightarrow}_0 \overset{*}{\twoheadrightarrow}_{>0}$, as for example we would not entail anything on a chain of the form $\rightarrow_0 \mapsto_{>0} \rightarrow_0$, which is in $\overset{*}{\twoheadrightarrow}$ but not in $(\twoheadrightarrow_0 \cup \twoheadrightarrow_{>0})^*$. This justifies the peculiar statement of the above lemma.

Theorem 21 (Standardization in DiLL). Take $\sim$ and $\overset{\mathsf{can}}{\rightarrow}$ to be exclusively $\overset{\mathsf{a}}{\sim}$ and $\overset{\mathsf{n}}{\rightarrow}$ respectively. Let π be a DiLL net. If $\pi \overset{*}{\twoheadrightarrow} \pi'$ (resp. $\pi \overset{\neg\epsilon^*}{\rightarrow} \pi'$), then there is a chain (called **standard**) of reductions and conversions (resp. of non-erasing reductions) from π to π' such that their depth is monotone increasing.

Proof. The proof goes by induction on the depth of π'. We can transform the given chain into $\pi \overset{*}{\twoheadrightarrow}_0 \nu \overset{*}{\twoheadrightarrow}_{>0} \pi'$ (resp. $\pi \overset{\neg\epsilon^*}{\rightarrow}_0 \nu \overset{\neg\epsilon^*}{\rightarrow}_{>0} \pi'$) by applying Lemma 20 (resp. Lemma 19). Such a chain induces a bijection f from $\mathsf{b}_0(\nu)$ to $\mathsf{b}_0(\pi')$ (the set of the 0-depth boxes), so that for every $c \in \mathsf{b}_0(\nu)$ we have $\sigma(c) \overset{*}{\twoheadrightarrow} \sigma(f(c))$ (the box contents), or resp. non-erasing steps. By inductive hypothesis there are standard chains taking all $\sigma(c)$ to $\sigma(f(c))$, and as these can be freely interleaved within ν, we end up with a standard chain from ν to π', which prepended with the 0-depth reductions from π to ν gives the desired standard chain of steps. □

Leaving out the general conversions and canonical steps is necessary, or else standardization fails. As an example take a cut c between the principal port of a box $!\pi$ and an auxiliary port of a box $!\xi$. Suppose that a reduction happening in ξ creates a contraction cell over the port corresponding to c. By a conversion we can push this contraction out of the box, so changing c into a contraction vs box cut. Performing one reduction step on c yields a net that cannot be obtained by the starting one by a standard reduction, as the only possible 0-depth step brings irreversibly $!\pi$ inside $!\xi$. Similar examples can be done using bang sum conversion and the two other canonical reductions. Notice that this drawback is not specific to DiLL but holds also in MELL.

4.2. *Dead Ends and their Manipulation*

In this section we introduce and study the rewriting $\rightsquigarrow$ (Figure 15), one of the two main tools used in the proof of Lemma 29 (the other being the unfolding defined in subsection 4.3).

We introduce in $\mathsf{DiLL}_{\partial\varrho}$ a new cell with one inactive port and no reduction rules, the **dead end** (graphically depicted by $\dashv$). As such cell is inert, it does not change anything of the computational properties of $\mathsf{DiLL}_{\partial\varrho}$, so all the results proved in section 3 are still valid. In order to manipulate such cells we introduce however a rewriting relation denoted

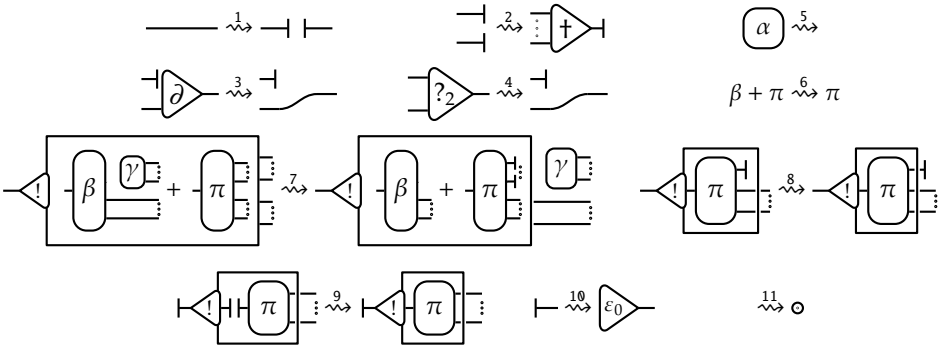


Figure 15. Rewriting rules for dead ends. † is any non-box cell, α is any simple net without free ports, β any simple net, γ any simple net not containing any ∂ or ϱ cells at depth 0, ε_0 a weakening or coweakening. Lax typing is omitted altogether as it is irrelevant for this rewriting.

by $\rightsquigarrow$ which is the context closure of the rules in Figure 15. Notice that $\rightsquigarrow$ does not necessarily preserve correctness, due to $\overset{2}{\rightsquigarrow}$. In the following we will prove the important properties of $\rightsquigarrow$.

Lemma 22. $\rightsquigarrow$ and $\overset{\neg\epsilon}{\to}$ commute, i.e. we have the following confluence diagram:

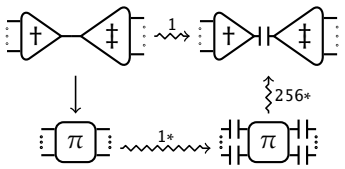
$$\begin{array}{ccc} \pi & \overset{*}{\rightsquigarrow} & \pi_1 \\ {\scriptstyle \neg\epsilon *}\downarrow & & \downarrow{\scriptstyle \neg\epsilon *} \\ \pi_2 & \overset{*}{\rightsquigarrow} & \pi_3 \end{array}$$

Proof. One proves that $\forall \pi, \pi_1, \pi_2, \exists \pi_3$:

$$\begin{array}{ccc} \pi & \rightsquigarrow & \pi_1 \\ {\scriptstyle \neg\epsilon}\downarrow & & \downarrow{\scriptstyle \neg\epsilon} = \\ \pi_2 & \overset{*}{\rightsquigarrow} & \pi_3 \end{array}$$

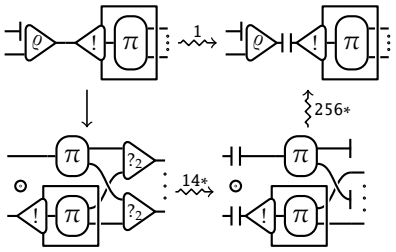
so that the result follows by Hindley's Lemma (Lemma 38). We here list the critical pairs, ordered by the $\rightsquigarrow$ rewriting rule in them.

1: The $\overset{1}{\rightsquigarrow}$ rewriting may interfere with a reduction by interrupting the cut to be reduced. Let us first tackle the case in which neither of the two interacting cells is a box. Then

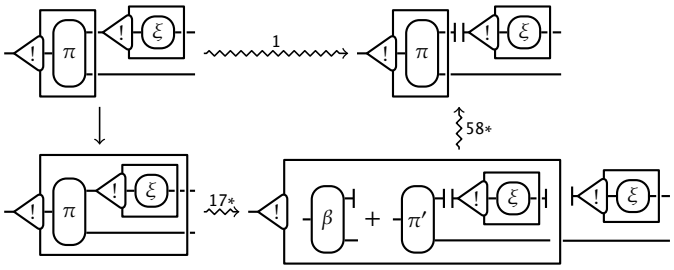


Here and in the following cases it is important that the reduction is non-erasing, so in

particular $\pi \neq 0$. This to assure there is enough to rebuild the redex or an immediate contractum. Step 6 may be necessary to delete addends added by the reduction.
Suppose now one of the two interacting cells is a box, so step 2 cannot be used to rebuild it. We have five cases, given by the other cell being ϱ, ∂, $?_2$, a box or a $!_2$.



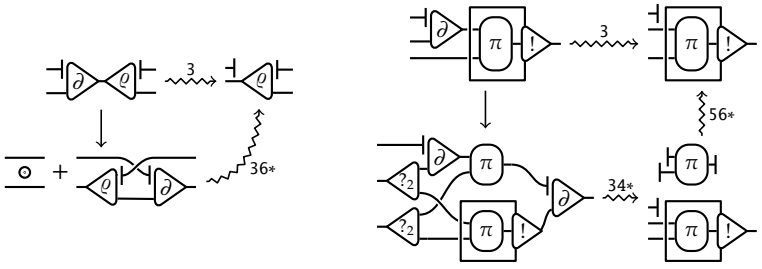
The cases for $?_2$ and ∂ are similar. As for box on box:



where $\pi = \beta + \pi'$ (as $\pi \neq 0$), and the number of wires is simplified. The case for $!_2$ is identical.

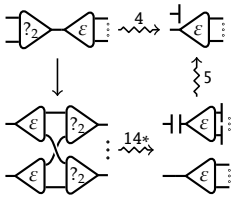
2: This step does not give rise to any critical pairs.

3: The critical pairs arise when the ∂ cell interacts with another cell. We show all but the case for a $?_2$-cell, as it is similar to the following ϱ one.



4: The case for a $?_2$ cell interacting with a ∂ cell is similar to the ∂/ϱ case above. Let ε

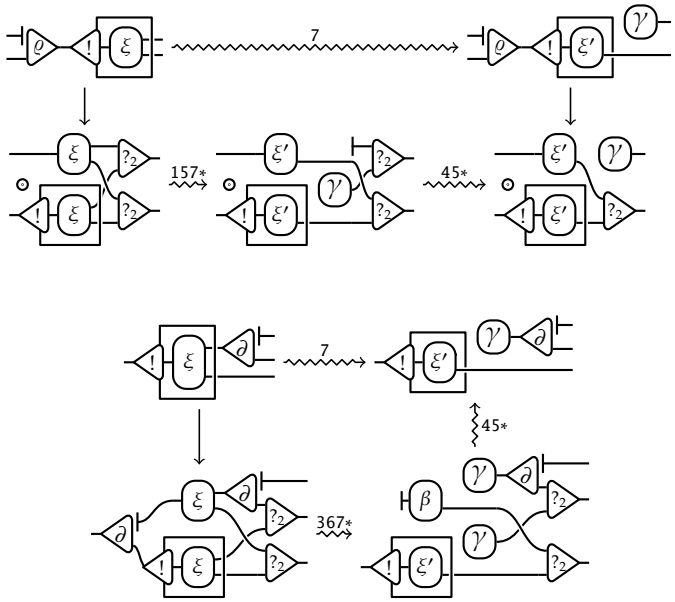
be either $!_2$ or $!\pi$, then:



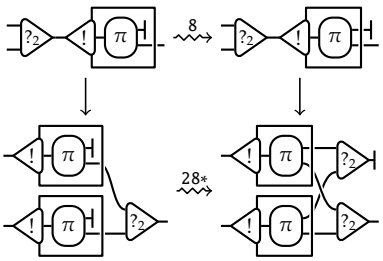
5,6: These steps pose no problem, as they interfere with a reduction only by completely deleting the location of the redex, which can be done also after firing it. In both cases more 6-steps could be necessary to erase additional addends spawned by the reduction.

7: First suppose the reduction takes place inside γ. As no 0-depth ∂ or ϱ cell is present in it, this reduction cannot spawn new addends, so the diagram is easily closed. If it takes place in β, then we can close by extracting γ from one of the addends originated by the reduction, and then eliminating all other instances of γ by means of 5-steps (as 7 severs their wires to the auxiliary ports). No difficulty arises when the reduction is in π.

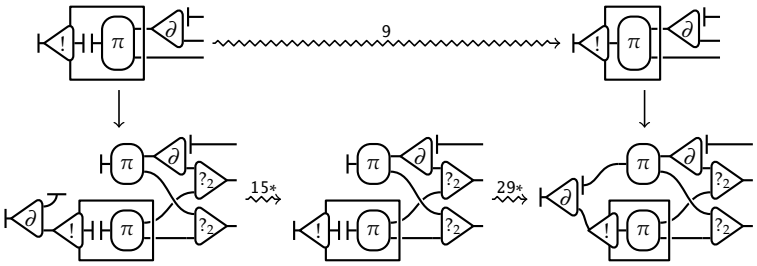
Let ξ be the contents of the box before the step, and ξ' afterwards with γ extracted and π's wires severed. Notice that $\xi \overset{15*}{\rightsquigarrow} \xi'$ up to dead ends over some of the free ports (corresponding to γ). We show the cases for $\varrho/!\xi$ and $\partial/!\xi$ with the ∂ cell on a port corresponding to γ. The other cases for ∂ and $?_2$ are very similar to the $\varrho/!\xi$ one, while $!_2$ and $!\rho$ on $!\xi$ are immediate (possibly gluing the entering cell to γ). We make a simplification on the number of wires, giving one free port to γ and one to β.



8: Any reduction happening in π, or $!_2$ and $!\rho$ cells entering the box pose no problems. The other cases are similar to the on for $?_2$:



9: As above, but the only case of relevance is $\partial/!\pi$:

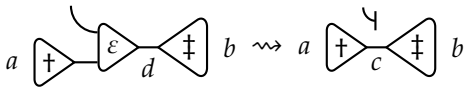


10,11: These steps have no critical pairs. □

Lemma 23. If $\pi \rightsquigarrow \xi$ and π is normal for $\neg\epsilon$-reduction, then so is ξ.

Proof. One reasons by cases on which rule of $\rightsquigarrow$ was applied. In fact the only cases that could give a problem are the ones destroying a ∂ or $?_2$ cell, as the other ones can not possibly introduce cuts.

Let c be a reducible cut in ξ that is not present in π. It must therefore be the wire resulting from one of the above mentioned $\rightsquigarrow$-reductions. The situation is thus the following:



with ε either ∂ or $?_2$, and with b neither a weakening nor a coweakening. b could be a 0 box only if $\dagger = ?_2$ and c is on the principal port. As c is reducible, and therefore not $\mho$-typed, so also d is not $\mho$-typed. It must be therefore the case that d is a reducible cut, so that π cannot be normal. □

Lemma 24. If π is $\mathsf{WN}_{\neg\epsilon}$ and $\pi \overset{*}{\rightsquigarrow} \xi$, then also ξ is $\mathsf{WN}_{\neg\epsilon}$.

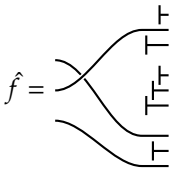
Proof. Direct consequence of Lemmata 22 and 23. □

4.3. *Unfolding*

Given any function $f : n \to k$ between finite ordinals[13], we define a net $\hat{f}$ which intuitively links n ports with k groups of n ports each, so that the i-th port among the n ones gets connected to the i-th port of the $f(i)$-th group, while the uncovered ports are severed by dead ends. Formally, let $\hat{f}$ be the net with:

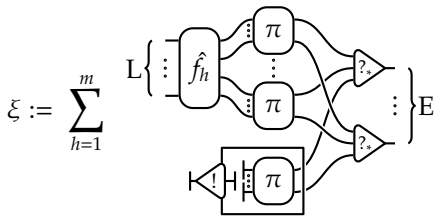
— free ports $\mathsf{fp}(\hat{f}) = n + kn$, denoted with $\{p_i\}_{i\in n} \cup \{q_{ji}\}_{j\in k, i\in n}$;
— cells $\mathsf{c}(\hat{f}) = \left\{ c^{ji} \mid j \in k, i \in n, j \neq f(i) \right\}$, all dead ends;
— wires $\mathsf{w}(\hat{f}) = \left\{ \{p_i, q_{f(i)i}\} \mid i \in n \right\} \cup \left\{ \{c_0^{ji}, q_{ji}\} \mid c^{ji} \in \mathsf{c}(\hat{f}) \right\}$, where c_0^{ij} denotes the sole port of the dead end c^{ij}.

For example take $f : 3 \to 3$ given by $f(0) = f(2) = 2$ and $f(1) = 0$. Then



We define a relation on pairs of contexts and polynets, called **unfolding** and denoted by $\omega[\pi] \twoheadrightarrow \omega[\xi]$. It holds when:

1. $\omega[\ ]$ is a linear context, with its hole interface partitioned into two interfaces L and E which we will call **linear** and **exponential**;
2. π is a polynet, with interface conveniently identified with $L + E$;
3. $\omega[\pi]$ is switching acyclic;
4. all wires passing through E in $\omega[\pi]$, with direction from π outward, are laxly typed by either ? or ⅋;
5. drawing L left and E right, ξ is



   with $m \geq 1$, and $f_h : \#L \to k_h$ for each h, with $k_h \geq 1$, where $?_*$ stands for any tree of contractions with a suitable number of leaves (namely $k_h + 1$ for the h-th addend);
6. for every switching path (see subsection 2.5) in $\omega[\ ]$ linking two ports of the hole both of its ends are in L, and moreover if these ends are the i-th and j-th ports of L, then $f_h(i) = f_h(j)$ for all addends of ξ, i.e. they are linked to the same linear copy of π.

We will sometimes blur the definition by seeing $\twoheadrightarrow$ as a relation between the polynets $\omega[\pi]$ and $\omega[\xi]$ rather than the pairs, as hinted by the notation. Notice that because of

[13] We recall that all natural numbers may be identified with the set of strictly less numbers, i.e. $n = \{0, \dots, n-1\}$.

the last point the relation is not contextual, even when restricting to linear contexts. Notice also that the last condition entails that also $\omega[\xi]$ is switching acyclic: it could fail to be so if a switching cycle was formed by passing through two auxiliary ports of the new box, but this can't happen by definition.

To get a first glimpse of what the above conditions mean, we can foretell we will use unfolding on the result of a $?/!\pi$ reduction: initially L will be the wire put in place of the principal port of $!\pi$, while the wires from the former auxiliary ports will form E. The last condition reflects what remains of the correctness criterion of boxes after they are opened. The shape of ξ is an evolution of what can be found after replacing the dereliction in $?/!\pi$ with its translation in $\mathsf{DiLL}_{\partial!}$ and reducing the resulting cut, which always leaves a copy of the box behind.

Lemma 25. If $\omega[\pi] \twoheadrightarrow \omega[\xi]$, then there are $\omega'[\ ]$, π', ξ' and ν such that

$$\begin{array}{ccc} \omega[\pi] & \longrightarrow\!\!\!\!\!\rightarrow & \omega[\xi] \\ \| & & \searrow^{\neg\epsilon *} \\ \omega'[\pi'] & \twoheadrightarrow & \omega'[\xi'] \overset{*}{\rightsquigarrow} \nu \end{array} \qquad (*)$$

where $\omega'[\pi']$ is a decomposition of $\omega[\pi]$ such that no wire connecting π' or ξ' to $\omega'[\ ]$ is a cut reducible under $\neg\epsilon$-reduction. Moreover if $\omega[\ ]$ is normal then so is $\omega'[\ ]$.

Proof. We reason by induction on the size of $\omega[\ ]$, proving that whenever there is a $\neg\epsilon$-reducible cut contradicting the thesis of the lemma, then there is $\omega'[\pi']$ satisfying diagram $(*)$ with $\omega'[\ ]$ smaller than $\omega[\ ]$ (in fact a subnet, so that the condition for a normal $\omega[\ ]$ follows). If we prove this we can in fact resolve the inductive step by

$$\begin{array}{ccccc} \omega[\pi] & \longrightarrow\!\!\!\!\!\rightarrow & \omega[\xi] & & \\ \| & & \searrow^{\neg\epsilon *} & & \\ \omega'[\pi'] & \twoheadrightarrow & \omega'[\xi'] \overset{*}{\rightsquigarrow} \nu & & \\ \| & & \searrow^{\neg\epsilon *} & \searrow^{\neg\epsilon *} & \\ \omega''[\pi''] & \twoheadrightarrow & \omega''[\xi''] \overset{*}{\rightsquigarrow} \nu' & \overset{*}{\rightsquigarrow} \nu'' & \end{array}$$

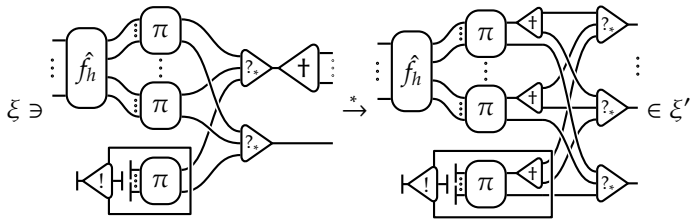
where $\omega''[\ ]$, π'' and ξ'' are obtained by inductive hypothesis with no cuts along the gluing, and the rightmost square is by Lemma 22.

First suppose the cut comes from a wire in $\omega[\ ]$ connecting two ports of the hole. By definition of $\omega[\pi] \twoheadrightarrow \omega[\xi]$, such wire must be on two ports in L which are connected both to the same copies of π in ξ. Define $\omega'[\ ]$ as $\omega[\ ]$ without such wire, and π' accordingly gluing the wire back. ξ' is immediately obtained from π' by following the choices made in ξ and just ignoring what was chosen for the missing ports of the interface. Then $\omega'[\xi']$ differs from $\omega[\xi]$ just by those copies of π in ξ that do not get the wire from $\omega[\ ]$. The diagram is then closed using several steps 1 of $\rightsquigarrow$ interrupting the glued wire where necessary. No $\neg\epsilon$-reduction of $\omega[\xi]$ is necessary.

Otherwise, if the cut is not from a wire in $\omega[\ ]$ on the hole, then at least one of the cells (say c) involved in the cut is in $\omega[\ ]$. We obtain $\omega'[\ ]$ and π' by detaching c from $\omega[\ ]$ and gluing it to π.

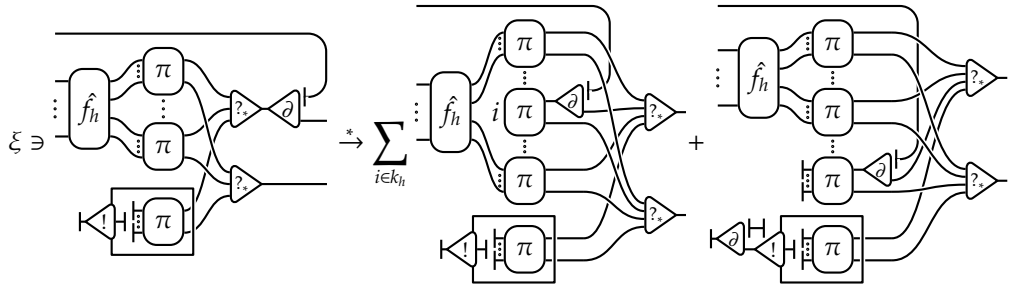
Suppose the said cut was in the linear interface L. The linear interface L' of $\omega'[\ ]$ has one less port (say p, the one on the cut) and some new ones due to the other ports of c. ξ' is obtained by following the same choices used for ξ, assigning to the new ports the same linear copies of π that were assigned to p. The only difference between $\omega[\xi]$ and $\omega'[\xi']$ is that all the copies that in $\omega[\xi]$ were not connected to p (including the one in the box), now have a copy of c, with dead ends on the ports not involved in the cut under examination. Steps 1 and 5 of $\rightsquigarrow$ easily bring $\omega'[\xi']$ back to $\omega[\xi]$, again with no needed $\neg\epsilon$-reduction. For showing $\omega'[\pi'] \twoheadrightarrow \omega'[\xi']$, now just the condition on switching paths needs to be checked (cond. 6). Take a switching path in $\omega'[\ ]$ linking two ports on the hole, exactly one among the new ones otherwise the condition is trivially met: if both are, then they are assigned to the same copies of π' by definition; if none are, it follows from $\omega[\pi] \twoheadrightarrow \omega[\xi]$. Notice that as p was on the cut, thus linked to an active port of c (be it a principal port or an auxiliary one of a box), such path induces one in $\omega[\ ]$ which is still switching and linking p, and the condition follows.

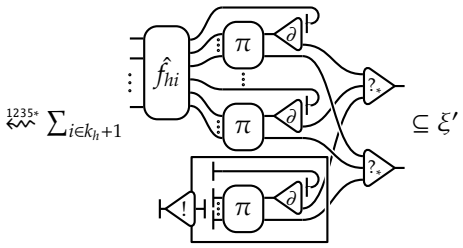
Suppose therefore that the cut is on the exponential interface E. By the lax type condition on $\omega[\pi]$ (cond. 4), for the cut to be reducible the cell c must be either a duplicable cell (i.e. a box or cocontraction) which we will denote by a generic symbol †, or a ∂ cell. In the first case, letting all the new ports of the hole of $\omega'[\ ]$ (i.e. the auxiliary ones of c) be in the exponential interface and defining ξ' by following the same choices made for ξ, we immediately get (depicting a single addend of ξ and ξ')



The condition on switching paths is met as † is always non-switching: no paths link its auxiliary ports (if any) among themselves or to other ports of the new interfaces.

In the latter case (i.e. c being a ∂ cell), we let one of the new ports be in the exponential interface and the other in the linear one and close the diagram in the following way (by appropriately defining ξ')





where if $f_h : n \to k_h$ then for $i \in k_h$ we set $f_{hi} : n \to k_h$ defined by $f_{hi}(0) = i$ and $f_{hi}(j+1) = f_h(j)$ (i.e. we complete f_h in all the possible ways for the new port), and finally $f_{hk_h} : n \to k_h + 1$ defined by $f_{hk_h}(0) = k_h$ and $f_{hk_h}(j+1) = f_h(j)$ (i.e. we further complete it by spawning a new linear copy of π). 3-steps take out the surplus of ∂ cells, letting 5-steps erase the resulting disconnected wires. A 1 and a 2 step are needed to recreate the ∂ cell linked to the box in the $k_h + 1$-th addend.

Again, the last thing to do is to check the condition on switching paths (cond. 6). No path links the new ports with any other port on the interfaces, otherwise $\omega[\ ]$ itself would not meet the condition. Moreover the two new ports are not linked between themselves as ∂ is non-switching. □

Lemma 26. If $\omega[\pi] \twoheadrightarrow \omega[\xi] \overset{*}{\rightsquigarrow} \nu$ and $\omega[\pi]$ is normal for $\neg\epsilon$-reduction, then $\nu \in \mathsf{WN}_{\neg\epsilon}$.

Proof. Using Lemmata 25 and 22 we get

$$\begin{array}{ccccc} \omega[\pi] & \twoheadrightarrow & \omega[\xi] & \overset{*}{\rightsquigarrow} & \nu \\ \| & & \searrow^{\neg\epsilon^*} & & \searrow^{\neg\epsilon^*} \\ \omega'[\pi'] & \twoheadrightarrow & \omega'[\xi'] & \overset{*}{\rightsquigarrow} & \nu' \quad \overset{*}{\rightsquigarrow} \nu'' \end{array}$$

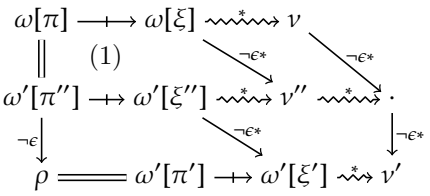
As $\omega[\pi] = \omega'[\pi']$ is normal, then so are $\omega'[\ ]$, π' and therefore ξ'. By the statement of Lemma 25 we deduce that $\omega'[\xi']$ is normal for $\neg\epsilon$-reduction, as such a statement explicitly forbids the plugging between ξ' and $\omega'[\ ]$ from forming new cuts. By Lemma 23 also ν'' is normal. □

Lemma 27. If $\omega[\pi] \twoheadrightarrow \omega[\xi] \overset{*}{\rightsquigarrow} \nu$, $\omega[\ ]$ is normal and $\omega[\pi] \overset{\neg\epsilon}{\to} \rho$, then there are $\omega'[\ ]$ normal, π', ξ' and ν' with

$$\begin{array}{ccccc} \omega[\pi] & \twoheadrightarrow & \omega[\xi] & \overset{*}{\rightsquigarrow} & \nu \\ {\scriptstyle\neg\epsilon}\downarrow & & & \searrow^{\neg\epsilon^*} & \\ \rho & = & \omega'[\pi'] & \twoheadrightarrow \omega'[\xi'] & \overset{*}{\rightsquigarrow} \nu' \end{array}$$

Proof. First of all we apply Lemma 25, getting $\omega'[\pi''] = \omega[\pi]$ and ξ'', with $\omega'[\ ]$ normal. As $\omega'[\pi''] \overset{\neg\epsilon}{\to} \rho$ but no cut is reducible along the gluing or in $\omega'[\ ]$, the reduction must happen in π''. We set $\pi' \overset{\neg\epsilon}{\leftarrow} \pi''$, by firing the same redex, so that clearly $\rho = \omega'[\pi']$. Moreover from $\pi'' \overset{\neg\epsilon}{\to} \pi'$ and ξ'' we get $\xi' \overset{\neg\epsilon^*}{\leftarrow} \xi''$ with $\omega'[\pi'] \twoheadrightarrow \omega'[\xi']$, by firing the

appropriate redexes in all copies made of π'' in its unfolding ξ''. Summing up:



where the diagram (1) comes from Lemma 25, as written above, and the whole diagram is closed using commutation (Lemma 22). The condition of normality of $\omega[\ ]$ is needed to prevent the context from additively duplicating its hole, and thus ceasing to be a context. □

It is now possible to prove the following lemma.

Lemma 28. If $\omega[\pi] \twoheadrightarrow \omega[\xi] \overset{*}{\rightsquigarrow} \nu$ and $\omega[\pi] \in \mathsf{WN}_{\neg\epsilon}$, then so is ν.

Proof. First, we reduce to the case where $\omega[\ ]$ is normal. By Lemma 10 (the conservation property for non-erasing reduction) $\omega[\pi] \in \mathsf{SN}_{\neg\epsilon\sim}$, and in particular $\omega[\ ] \in \mathsf{WN}_{\neg\epsilon}$. Reducing the context to normal form for $\neg\epsilon$-reduction we get $\omega[\pi] \xrightarrow{*} \sum_i \omega_i[\pi]$ and $\omega[\xi] \xrightarrow{*} \sum_i \omega_i[\xi]$ with $\omega_i[\ ]$ normal: as $\omega[\ ]$ is linear the hole may be duplicated only additively. We see that for each i we have $\omega_i[\pi] \twoheadrightarrow \omega_i[\xi]$, as reduction does not create switching paths (Lemma 4), so the last condition is met. Moreover $\omega_i[\pi]$ must still be $\mathsf{SN}_{\neg\epsilon\sim}$ (therefore $\mathsf{WN}_{\neg\epsilon}$) and, by Lemma 22, $\nu \xrightarrow{\neg\epsilon^*} \sum_i \nu_i$ with $\omega_i[\xi] \overset{*}{\rightsquigarrow} \nu_i$. If we have $\nu_i \in \mathsf{WN}_{\neg\epsilon}$ for all i we are done, so we have reduced the proof to the case where $\omega[\ ]$ is normal.

By the conservation theorem for $\mathsf{DiLL}_{\partial\varrho}$ we are also entitled to reason by well-founded induction on $\omega[\pi]$. The base case for $\omega[\pi]$ normal is in fact Lemma 26, while if $\omega[\pi] \to \rho$ we can now apply Lemma 27 and then inductive hypothesis on $\omega'[\pi']$ and get the result. □

4.4. *The Main Lemma*

We are finally ready to prove the main lemma enabling to transfer the conservation theorem from $\mathsf{DiLL}_{\partial\varrho}$ to DiLL via the translation π^Δ.

Lemma 29. Let π be a pure DiLL switching acyclic. If $\pi \in \mathsf{WN}_{\neg\epsilon}$ then $\pi^\Delta \in \mathsf{WN}_{\neg\epsilon}$ in $\mathsf{DiLL}_{\partial\varrho}$.

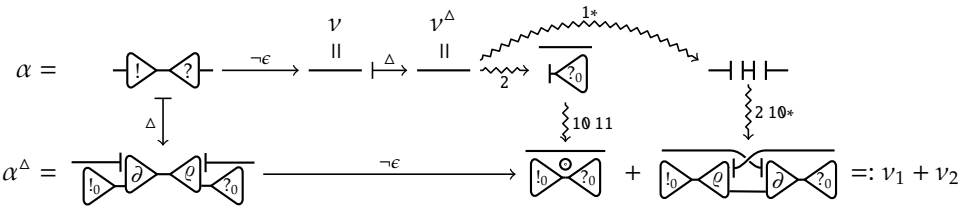
Proof. First, using Theorem 21, consider a standard normalizing reduction chain R of π to normal form. Let us reason by induction on (n, d) where n is the length of R and d is the depth of the first reduction step (i.e. the minimum depth of all R).

If $n = 0$, i.e. π is normal for $\xrightarrow{\neg\epsilon}$, then so is π^Δ. If both $n > 0$ and $d > 0$ then let $\nu_1, \ldots, \nu_k$ be all the contents of the boxes at depth 0 in π. As the reduction is standard, π (and therefore π^Δ) is normal at depth 0 and all R can be partitioned into standard chains R_i reducing each ν_i to normal form. Now the inductive hypothesis kicks in: if the partition

is trivial (i.e. the reduction happens all in one ν_i) then R_i has the same length but lesser depth; if it is not trivial then each R_i has strictly less reductions. In any case ν_i^Δ are $\mathsf{WN}_{\neg\epsilon}$, and following such normalizing reductions in π^Δ also leads to a normal form.

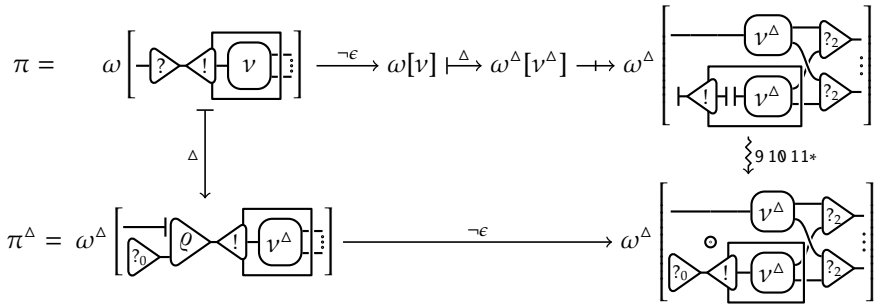
Suppose therefore $n > 0$ and $d = 0$, so that $\pi = \omega[\alpha] + \rho \xrightarrow{\neg\epsilon} \omega[\nu] + \rho$ with α and ν a redex/contractum pair, $\omega[\ ]$ a linear context, and by inductive hypothesis $(\omega[\nu] + \rho)^\Delta = \omega^\Delta[\nu^\Delta] + \rho^\Delta \in \mathsf{WN}_{\neg\epsilon}$. For every reduction which does not touch derelictions or coderelictions, the inductive step is immediate, as the translation in $\mathsf{DiLL}_{\partial!}$ leaves the reduction unchanged apart from possibly differing on pebbles. We thus have that $\pi^\Delta \xrightarrow{\neg\epsilon} \overset{11*}{\rightsquigarrow} (\omega[\nu]+\rho)^\Delta$, so that Lemma 24 yields what needed. Let us see all the remaining cases. We will drop ρ, as closing by sum is trivial.

Dereliction on codereliction:



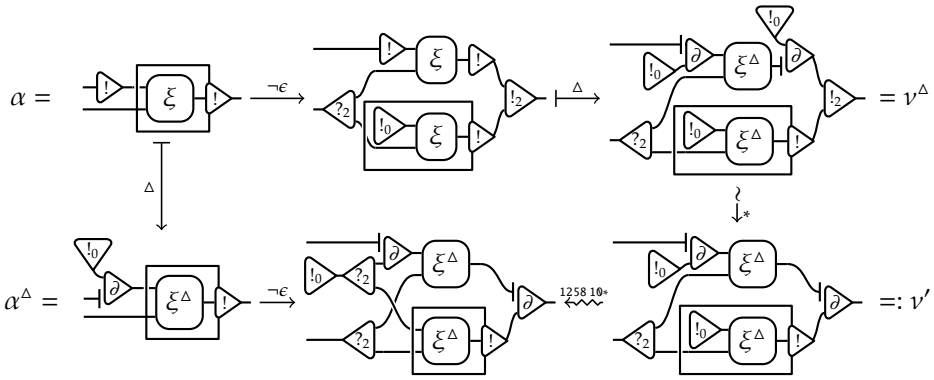
As $\omega^\Delta[\nu^\Delta]$ is $\mathsf{WN}_{\neg\epsilon}$ and $\omega^\Delta[\nu^\Delta] \overset{*}{\rightsquigarrow} \omega^\Delta[\nu_i]$ for both i, it follows from Lemma 24 that both $\omega^\Delta[\nu_i] \in \mathsf{WN}_{\neg\epsilon}$, so $\pi^\Delta \xrightarrow{\neg\epsilon} \omega^\Delta[\nu_1] + \omega^\Delta[\nu_2]$ is also.

(Co)dereliction on co(co)ntraction :



Notice that $\omega^\Delta[\nu^\Delta]$ satisfies the requirements of $\twoheadrightarrow$ as π and π^Δ are switching acyclic. Lemma 28 applied to $\omega^\Delta[\nu^\Delta] \in \mathsf{WN}_{\neg\epsilon}$ directly gives the result.

Codereliction on box:



By inductive hypothesis, $\omega^{\Delta}[\nu^{\Delta}]$ is $\mathsf{WN}_{\neg\epsilon}$, so by Theorem 17 we have $\omega^{\Delta}[\nu^{\Delta}] \in \mathsf{SN}_{\sim}$. This implies that $\omega^{\Delta}[\nu']$, obtained by an associative conversion and a neutral reduction, is also $\mathsf{SN}_{\sim}$ and $\mathsf{WN}_{\neg\epsilon}$. Lemma 24 therefore gives the desired result. □

Theorem 30 (Conservation for DiLL). Given a net π in DiLL, if π is switching acyclic and $\pi \in \mathsf{WN}_{\neg\epsilon}$, then $\pi \in \mathsf{SN}_{\sim}$.

Proof. The proof is now immediate. Take a DiLL switching acyclic net π. Then

$$\begin{aligned}
\pi \in \mathsf{WN}_{\neg\epsilon} &\implies \pi^{\Delta} \in \mathsf{WN}_{\neg\epsilon} && \text{by Lemma 29;}\\
&\implies \pi^{\Delta} \in \mathsf{SN}_{\sim} && \text{by Theorem 17, Conservation for } \mathsf{DiLL}_{\partial!};\\
&\implies \pi \in \mathsf{SN}_{\sim} && \text{by Lemma 18.}
\end{aligned}$$

□

5. An Application: $\mathsf{SN}_{\sim}$ of Simply Typed DiLL and $\mathsf{DiLL}_{\partial!}$

We introduce simply typed DiLL and $\mathsf{DiLL}_{\partial!}$: both systems can be proved to be $\mathsf{WN}_{\neg\epsilon}$ by adapting the technique from [Pag09], therefore conservation yields $\mathsf{SN}_{\sim}$. We define simple typing as a type assignment characterizing a subclass of laxly typed nets. Indeed we here move from the assumption that the syntactic objects we are dealing with are inherently laxly typed as we defined them at the beginning of this paper. However we will assure that the class of simply typable nets is closed under $\twoheadrightarrow$ and that in fact lax types can be forgotten altogether (Proposition 31), so that the simply typed system *is* the one expected. Analogously one may apply the conservation theorem also on second order DiLL, which is not presented here.

Simple types of Linear Logic are given by the following grammar, where X ranges over a fixed set of propositional variables:

$$A, B ::= X \mid X^{\perp} \mid 1 \mid \bot \mid A \otimes B \mid A \parr B \mid !A \mid ?A.$$

Linear negation is an involution ($A^{\perp\perp} = A$) defined via the De Morgan laws: $(X)^{\perp} := X^{\perp}$, $(A \otimes B)^{\perp} := A^{\perp} \parr B^{\perp}$, $(!A)^{\perp} := ?A^{\perp}$. There is a function U (a sort of forgetful map) from simple types to lax ones, given by $U(X) = U(X^{\perp}) = \Omega$, $U(c) = c$, $U(A \square B) = \square$ and

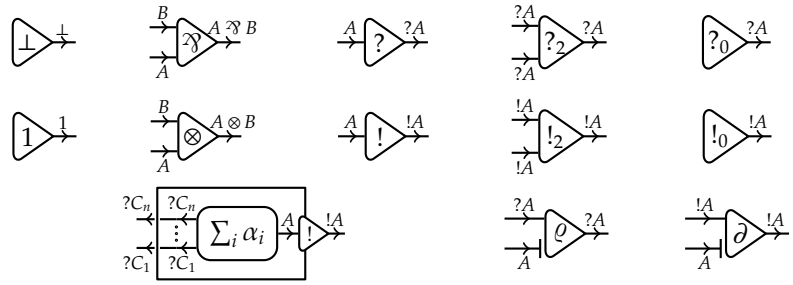


Figure 16. Simple typing for DiLL and $\mathsf{DiLL}_{\partial\varrho}$.

$U(\diamond A) = \diamond$ for c, $\square$ and $\diamond$ one of the two units 1 and $\bot$, the two connectives $\otimes$ and $⅋$ and the two exponential modalities ! and ? respectively.

A differential net π is **simply typable** whenever there is a function ℓ from the oriented wires of π to the set of simple types respecting the conditions of Figure 16, such that opposite directions of a wire are associated with dual types and such that the underlying lax typing ℓ' enjoys $\ell'(d) = U(\ell(d))$. So the lax typing can be recovered from the simple one, and indeed the rules for simple typing of Figure 16 assure that also the ones for lax typing (Figures 2 and 9) are satisfied by $U \circ \ell$. Notice that in particular this means that a simply typed net has no $\mathfrak{O}$-typed wires. Notice also that both the translations $(\,)^{\triangle}$ and $(\,)^{\triangledown}$ from DiLL to $\mathsf{DiLL}_{\partial\varrho}$ and back preserve simple typability.

Proposition 31. In DiLL and $\mathsf{DiLL}_{\partial\varrho}$ if $\pi \to \rho$ (resp. $\pi \mapsto \rho$) and π is simply typable then so is ρ.

Proof. Easy case inspection. In fact given any rule $\alpha \to \nu$ or $\alpha \mapsto \nu$ and a simple typing ℓ of α, it is straightforward to define a typing ℓ' of ν preserving the ℓ types on terminal wires. Plugging in the context then gives the result. □

The above property means also that $\mathfrak{O}$ never arises in a reduction starting from a simply typed net so that no cut gets blocked by lax typing (in particular banning clashes). As the only *raison d'être* of lax typing is therefore missing, we can altogether forget it.

Proposition 32. In DiLL simply typed and switching acyclic differential nets are $\mathsf{WN}_{\neg\epsilon}$.

Proof (Sketch). We adapt to non-erasing reduction and somewhat simplify (at the cost of using a stronger theorem along the way) the proof of WN given in [Pag09] for DiLL.

Let the grade of a wire in π be the size of its type, i.e. the number of connectives in the formula, which is invariant under linear negation. We can then use the finite developments theorem from [Tra09a] to obtain that, given a fixed grade, reducing only non-erasing cuts with that grade is necessarily strongly normalizing. Indeed there are two types of "new" cuts blocked by the definition of developments: on one side there are clashes, that are of no concern here, on the other are cuts whose type is necessarily "simpler" (i.e. of strictly less grade) than the one fired. So the reduction reducing cuts of a fixed grade

is a development (i.e. it is a reduction of the system DiLL° following the terminology of [Tra09a]) and is therefore terminating.

A round-by-round normalizing strategy is now easy to define: one reduces non-erasing cuts in decreasing order of grade, in any order for cuts of equal grade. The maximum grade of the net then decreases strictly at each round involving the cuts with that grade. □

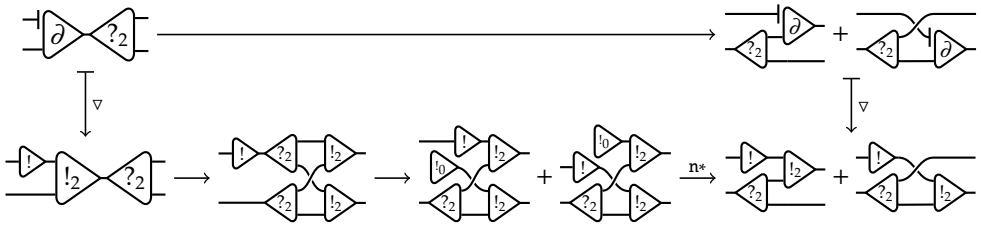
Theorem 33. In DiLL simply typed and switching acyclic differential nets are $\mathsf{SN}_\sim$.

Proof. Consequence of Proposition 32 and the conservation theorem. □

We can achieve $\mathsf{SN}_\sim$ of simply typed $\mathsf{DiLL}_{\partial\varrho}$ following two different ways: either we prove $\mathsf{WN}_{\neg\epsilon}$ using again a variant of [Pag09] and then we apply $\mathsf{DiLL}_{\partial\varrho}$ conservation, or we use the translation $(\ )^\nabla$ for simulating $\mathsf{DiLL}_{\partial\varrho}$ by means of DiLL (which is needed anyway to obtain the finite developments theorem for $\mathsf{DiLL}_{\partial\varrho}$). Let us follow this last direction, the former presenting no new interesting point. The following lemma is the $\mathsf{DiLL}_{\partial\varrho}$ counterpart of Lemma 18.

Lemma 34 (DiLL simulates $\mathsf{DiLL}_{\partial\varrho}$). If in $\mathsf{DiLL}_{\partial\varrho}$ $\pi \rightsquigarrow \rho$ then $\pi^\nabla \xrightarrow{+}\!\!\!\!\rightsquigarrow \rho^\nabla$ in DiLL. In particular, if π^∇ is $\mathsf{SN}_\sim$, then so is π in $\mathsf{DiLL}_{\partial\varrho}$.

Proof. The only interesting cases are the ones involving one ϱ or ∂ cell. We leave all but one as an exercise for the reader. Let us detail the linear substitution on contraction case:



□

Theorem 35. In $\mathsf{DiLL}_{\partial\varrho}$, simply typed and switching acyclic differential nets are $\mathsf{SN}_\sim$.

Proof. Consequence of Lemma 34 and Theorem 33, recalling that π^∇ is simply typed and switching acyclic whenever π so is. □

References

[Bar81] Henk Barendregt. *The lambda calculus, its syntax and semantics*. Number 103 in Studies in Logic and the Foundations of Mathematics. North-Holland, first edition, 1981.

[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.

[DCG99] Roberto Di Cosmo and Stefano Guerrini. Strong normalization of proof nets modulo structural congruences. *LNCS*, 1631:75–89, 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.

[DCKP03] Roberto Di Cosmo, Delia Kesner, and Emmanuel Polonovski. Proof nets and explicit substitutions. *Mathematical Structures in Comp. Sci.*, 13(3):409–450, jun 2003.

[dF09] Marc de Falco. An explicit framework for interaction nets. In Ralf Treinen, editor, *RTA*, volume 5595 of *Lecture Notes in Computer Science*, pages 209–223. Springer, 2009.

[DR89] Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.

[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.

[ER06] Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theor. Comput. Sci.*, 364(2):166–195, 2006.

[Fio07] Marcelo P. Fiore. Differential structure in models of multiplicative biadditive intuitionistic linear logic. In Simona Ronchi Della Rocca, editor, *TLCA*, volume 4583 of *Lecture Notes in Computer Science*, pages 163–177. Springer, 2007.

[Gan80] R. O. Gandy. Proofs of strong normalization. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 457–477. Academic Press Limited, 1980.

[Gir72] Jean-Yves Girard. *Interprétation Fonctionnelle et Élimination des Coupures de l'Arithmétique d'Ordre Supérieur.* Thèse de doctorat d'etat, Université Paris VII, 1972.

[Gir87a] Jean-Yves Girard. Linear Logic. *Th. Comp. Sc.*, 50:1–102, 1987.

[Gir87b] Jean-Yves Girard. *Proof Theory and Logical Complexity.* Studies in Proof-theory. Bibliopolis, Napoli, 1987.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types.* Number 7 in Cambridge tracts in theoretical computer science. Cambridge University Press, 1989.

[Hin64] J Roger Hindley. *The Church-Rosser property and a result in combinatory logic.* PhD thesis, University of Newcastle-upon-Tyne, 1964.

[Hue80] Gérard P. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *J. ACM*, 27(4):797–821, 1980.

[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.

[Ned73] Robert Pieter Nederpelt. *Strong Normalization for a Typed Lambda Calculus with Lambda Structured Types.* Ph.D. thesis, Technische Hogeschool Eindhoven, 1973.

[Pag09] Michele Pagani. The cut-elimination theorem for differential nets with promotion. In Pierre-Louis Curien, editor, *TLCA*, volume 5608 of *Lecture Notes in Computer Science*, pages 219–233. Springer, 2009.

[PTdF10] Michele Pagani and Lorenzo Tortora de Falco. Strong Normalization Property for Second Order Linear Logic. *Theoretical Computer Science*, (411):410–444, 2010. Doi: 10.1016/j.tcs.2009.07.053.

[Sør97] Morten Heine Sørensen. Strong Normalization from Weak Normalization in Typed λ-Calculi. *Inf. Comput.*, 133(1):35–71, 1997.

[Tai67] William W. Tait. Intensional interpretation of functionals of finite type. *Journal of Symbolic Logic*, 32(2):187–199, 1967.

[TdF03] Lorenzo Tortora de Falco. Additives of linear logic and normalization- part I: a (restricted) Church-Rosser property. *Theoretical Computer Science*, 294/3:489–524, 2003.

[Ter03] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

[Tra08] Paolo Tranquilli. Intuitionistic differential nets and lambda calculus. Theor. Comput. Sci., to appear, 2008.

[Tra09a] Paolo Tranquilli. Confluence of pure differential nets with promotion. In Erich Grädel and Reinhard Kahle, editors, *CSL*, volume 5771 of *Lecture Notes in Computer Science*, pages 500–514. Springer, 2009.

[Tra09b] Paolo Tranquilli. *Nets between determinism and nondeterminism*. Ph.D. thesis, Università Roma Tre/Université Paris Diderot (Paris 7), April 2009.

[Vau07] Lionel Vaux. *λ-calcul différentiel et logique classique : interactions calculatoires*. Thèse de doctorat, Université de la Méditerranée, 2007.

Appendix A. Compendium of Rewriting Theory

The aim of this appendix is making the reader acquainted with the notations and notions of rewriting and rewriting modulo that we employ in this paper. We refer to [Ter03] for a more in-depth presentation complete with proofs. Let us note however that Lemmata 37 and 39 are new results, as far as we know.

Let $(A, \to)$ be an abstract reduction system. The relations $\xrightarrow{=}$, $\xrightarrow{+}$, $\xrightarrow{*}$, $\leftarrow$ are respectively the reflexive, transitive, reflexive-transitive closures and the transpose of $\to$. The composition of two relations R and T is denoted by juxtaposition, so that $s\ RT\ t$ iff $\exists u$ with $s\ R\ u$ and $u\ T\ s$. A term t is normal if there is no u with $t \to u$; t is weakly normalizable ($\mathsf{WN}_\to$) whenever there is u normal such that $t \xrightarrow{*} u$; t is strongly $\to$-normalizable ($\mathsf{SN}_\to$) whenever there is no infinite sequence $(t_i)_{i\in\mathbb{N}}$ such that $t_0 = t$ and $t_i \to t_{i+1}$. We say that $\to$ is weak (resp. strongly) normalizable if so is every term of A. We may avoid specifying $\to$ in $\mathsf{WN}_\to$ and $\mathsf{SN}_\to$ whenever it is clear which relation we refer to.

Let $\mapsto$ be a symmetric relation on A generating an equivalence $\sim$, i.e. $\sim = \mapsto^*$ (possibly $\mapsto$ can be $\sim$ itself). We denote by $\rightsquigarrow$ the concatenation $\sim\!\!\to\!\!\sim$. We speak of weak/strong normalization modulo $\sim$ ($\mathsf{WN}_{\to\sim}$ and $\mathsf{SN}_{\to\sim}$ for short) intending the corresponding notion of weak/strong normalization of $\rightsquigarrow$. We say that (see Figure 17) $\to$ is confluent modulo $\sim$ if $\xleftarrow{*}\!\!\sim\!\!\xrightarrow{*} \subseteq \xrightarrow{*}\!\!\sim\!\!\xleftarrow{*}$; Church-Rosser modulo $\sim$ (or $\mathsf{CR}\sim$) if $(\to \cup \leftarrow \cup \sim)^* \subseteq \xrightarrow{*}\!\!\sim\!\!\xleftarrow{*}$; locally confluent modulo $\sim$ if $\leftarrow\!\!\to \subseteq \xrightarrow{*}\!\!\sim\!\!\xleftarrow{*}$; locally coherent with $\mapsto$ if $\mapsto\!\!\to \subseteq \xrightarrow{*}\!\!\sim\!\!\xleftarrow{*}$.

Clearly by setting $\sim$ to be the identity on terms one recovers the usual terminology of (local) confluence and Church-Rosser. Notice however that being Church-Rosser modulo $\sim$ is strictly stronger than plain confluence in absence of WN [Ter03, Remarks 14.3.6, Exercise 14.3.7].

We will use the following important generalization of the Newman lemma.

Lemma 36 (Huet, [Hue80]). Let $(A, \to, \mapsto)$ be an abstract reduction system, $\mapsto$ being a symmetric relation generating the equivalence $\sim$. If $\to$ is $\mathsf{SN}_\sim$, locally coherent with $\mapsto$ and locally confluent modulo $\sim$, then it is $\mathsf{CR}\sim$.

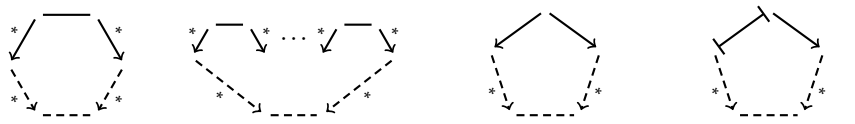


Figure 17. Various confluence patterns. A straight line indicates the equivalence $\sim$. In local coherence $\vdash$ generates $\sim$, i.e. $\sim = \vdash^*$.

The following lemma extends to this setting item (iii) of Theorem 1.2.3 at page 18 in [Ter03].

Lemma 37 (normalization modulo by increasing measure). Let $(A, \to, \vdash)$ be an abstract reduction system, $\vdash$ being a symmetric conversion step generating an equivalence $\sim$. Suppose

1. $\to$ is locally confluent modulo $\sim$,
2. $\to$ is locally coherent with $\vdash$,
3. there is a measure $|\cdot| : A \to \mathbb{N}$ that is strictly increasing under $\to$ and invariant under $\sim$.

Then $t \in \mathsf{WN}$ implies $t \in \mathsf{SN}_\sim$.

Proof. First notice that 2 and 3 entail

$$\text{every term equivalent to a normal form is a normal form.} \tag{2}$$

In fact suppose $t_0 \vdash t$ for a normal form t_0. If t is not a normal form, there is a term s s.t. $t \to s$. By local coherence there is $s' \sim t_0$ s.t. $s \xrightarrow{*} s'$. By 3 we have $|t_0| = |t| < |s| \leq |s'| = |t_0|$, which is absurd. Therefore both $\vdash$ and $\sim = \vdash^*$ preserve being a normal form. In particular, all normal forms are also $\mathsf{SN}_\sim$, which need not be in general.

We will show that if $t \xrightarrow{*} t_0$ with t_0 normal then $t \in \mathsf{SN}_\sim$, reasoning by induction on $|t_0| - |t|$, which is positive by 3. If it is 0, then necessarily $t = t_0$ and by (2) $t \in \mathsf{SN}_\sim$. Suppose therefore that it is strictly positive, which implies $t \to t_1 \xrightarrow{*} t_0$ with $t_1 \in \mathsf{SN}_\sim$ by inductive hypothesis.

First, we see that if $t \vdash s$ then there is a normalizing chain $s \xrightarrow{*} s_0$ with $t_0 \sim s_0$. The situation is indeed the following.

$$\begin{array}{ccccc} t & \longrightarrow & t_1 & \xrightarrow{*} & t_0 \\ \mathrel{\rotatebox{90}{$\vdash$}} & \text{(i)} & \mathrel{\rotatebox{90}{$\dashrightarrow$}}{}^{*} & \text{(ii)} & \mathrel{\rotatebox{90}{$\dashrightarrow$}} \\ s & \longrightarrow & s_1 & \xrightarrow{*} & s_0 \end{array}$$

(i) is local coherence modulo, from item 2, while (ii) is entailed by $\mathsf{CR}_\sim$ on the $\twoheadrightarrow$-descendants of t_1, which is in turn given by items 1 and 2 and applying Lemma 36 as $t_1 \in \mathsf{SN}_\sim$. Notice that as $|s_0| - |s| = |t_0| - |t|$ (item 3), we can iterate the above and get the same if starting from $t \sim s$.

Now take any s such that $t \twoheadrightarrow s$, i.e. $t \sim t' \to s' \sim s$. By the above we get $t' \to t'_1 \xrightarrow{*}$

$t'_0 \sim t_0$ with $|t'_0| - |t'| = |t_0| - |t|$ (with t'_1 existing as the latter is strictly positive). Now we can repeat the above confluence diagram with t' and s' in place of t and s, and using local confluence modulo (item 1) rather than local coherence, to get $s' \xrightarrow{*} s'_0 \sim t_0$. As $|s'_0| - |s'| < |t'_0| - |t'|$, inductive hypothesis gives $s' \in \mathsf{SN}_\sim$ which in turn entails that $s \in \mathsf{SN}_\sim$ also. As all $\rightsquigarrow$ steps from t brings inside $\mathsf{SN}_\sim$, then necessarily $t \in \mathsf{SN}_\sim$. □

Let $(A, \xrightarrow{1}, \xrightarrow{2})$ be an abstract reduction system. The reduction $\xrightarrow{12}$ is the union of reductions $\xrightarrow{1}$ and $\xrightarrow{2}$. The following is in Hindley's PhD Thesis, it can be also found in [Ter03, Exercise 1.3.6].

Lemma 38 (Hindley, [Hin64]). Let $(A, \xrightarrow{1}, \xrightarrow{2})$ be an abstract reduction system such that $\forall u, t, v \in A, u \xleftarrow{1} t \xrightarrow{2} v$ implies $\exists z \in A, u \xrightarrow{2*} z \xleftarrow{1=} v$. Then $\xrightarrow{1}$ and $\xrightarrow{2}$ commute, i.e. $\xleftarrow{1*}\xrightarrow{2*} \subseteq \xrightarrow{2*}\xleftarrow{1*}$.

Lemma 39. Let $(A, \xrightarrow{1}, \xrightarrow{2})$ be an abstract reduction system such that there is a measure $|\cdot| : A \to \mathbb{N}$ with $|x| > |y|$ (resp. $|x| \geq |y|$) whenever $x \xrightarrow{1} y$ (resp. $x \xrightarrow{2} y$), and such that $\xrightarrow{1}\xrightarrow{2} \subseteq \xrightarrow{12*}\xrightarrow{1}$. Then $\xrightarrow{1}$ can be postponed with respect to $\xrightarrow{2}$, i.e. $\xrightarrow{12*} \subseteq \xrightarrow{2*}\xrightarrow{1*}$.

Proof. Take $x \xrightarrow{12*} y$, and let us reason by induction on $|x| - |y|$, which by hypothesis is a positive integer. Suppose in the reduction chain there is at least a 1-reduction, or else the result is trivial. We then have $x \xrightarrow{12*}\xrightarrow{1}\xrightarrow{2*} y$: we can turn it into $x \xrightarrow{12*} y' \xrightarrow{1} y$ iterating the hypothesis $\xrightarrow{1}\xrightarrow{2} \subseteq \xrightarrow{12*}\xrightarrow{1}$ (formally reasoning by a secondary induction on the length of the final 2-reduction). Now $|x| - |y'| < |x| - |y|$, so by inductive hypothesis we get $x \xrightarrow{2*}\xrightarrow{1*} y' \xrightarrow{1} y$ which concludes the proof. □

Notice that the above lemma may be employed dually, i.e. if $|.|$ increases strictly (resp. does not decrease) for $\xrightarrow{1}$ (resp. for $\xrightarrow{2}$) and $\xrightarrow{2}\xrightarrow{1} \subseteq \xrightarrow{1}\xrightarrow{12*}$ then $\xrightarrow{2}$ can be delayed with respect to $\xrightarrow{1}$: it suffices to use the lemma on the expansions $\xleftarrow{1}$ and $\xleftarrow{2}$, transposing all the equations on the relations.