

Strong Normalization property for second order Linear Logic

Michele Pagani ¹, Lorenzo Tortora de Falco

*Dipartimento di Filosofia – Università Roma Tre
Via Ostiense 231, 00144 Roma, Italy*

Abstract

The paper contains the first complete proof of strong normalization (SN) for full second order linear logic (LL): Girard's original proof uses a standardization theorem which is not proven.

We introduce sliced pure structures (sps), a very general version of Girard's proof-nets, and we apply to sps Gandy's method to infer SN from weak normalization (WN). We prove a standardization theorem for sps: if WN without erasing steps holds for an sps, then it enjoys SN. A key step in the proof of standardization is a confluence theorem for sps obtained by using only a very weak form of correctness, namely acyclicity slice by slice.

We conclude by showing how standardization for sps allows to prove SN of LL, using as usual Girard's reducibility candidates.

Key words: (weak and strong) normalization, confluence, linear logic, standardization, sliced pure structures, proof-nets

1 Introduction

In every abstract approach to computation, the distinction between terminating and non-terminating processes is crucial. A rewriting system enjoys *weak normalization* (WN) if every term of the system can be executed in a finite number of steps.

In the λ -calculus (by far the most studied rewriting system), non terminating computations start from λ -terms that strongly exploit *self-application*: every λ -term can

Email addresses: pagani@uniroma3.it (Michele Pagani),
tortora@uniroma3.it (Lorenzo Tortora de Falco).

¹ Supported by postdoc fellowship "Ricerche sulla geometria della logica", Dipartimento di Filosofia, Università Roma Tre

be applied to itself (see for example [Kri90]). Termination fails for the λ -calculus (even in its weak form WN), but holds for its notable subsystems the simply typed λ -calculus and its extension Girard's system F ([Gir72]). Behind these termination proofs hides a deep connection between WN and consistency (in the logical sense) established by the Curry-Howard correspondence between proofs and λ -terms (a proof is a λ -term whose execution corresponds to applying the cut-elimination procedure to the proof). A λ -term associated with a proof π can be executed in such a way that a result is reached, when π can be transformed into a cut-free proof (that is a proof without lemmata): WN corresponds exactly to cut-elimination. In traditional proof-theory (dating back to Gentzen) cut-elimination is a key property of a logical system, from which the consistency of the system immediately follows²: WN of the simply typed λ -calculus (resp. of system F) corresponds then to a consistency proof for natural deduction ND (resp. second order natural deduction ND_2).

The Curry-Howard approach is a way to put constraints on the possibility of building λ -terms, and more precisely on self-application: from the logical point of view, these constraints are sufficient conditions to prove the consistency of the corresponding logical system³. While in the simply typed λ -calculus self-application is simply forbidden, in system F a (weak) form of self-application is accepted. System F combines then computational strength and termination (here is the remarkable interest of WN for F). The most famous result of mathematical logic (Gödel's second incompleteness theorem) sheds then a new light on the "difficulty" (and the deep meaning) of the termination property. Indeed, Peano Arithmetic PA can be translated in ND_2 , so that WN for F cannot be proven within ND_2 (and of course this holds for any subsystem of the λ -calculus whose corresponding logical system contains -a copy of- PA).

The terms of system F actually enjoy *strong normalization* (SN), a much stronger termination property than WN: whatever execution strategy one applies to F's terms one eventually reaches a result (a normal form). This computationally relevant variant of WN has a mathematical counterpart: it corresponds to saying that the tree of the computations starting from a term is well-founded. Several techniques to infer SN from WN have been proposed (see for example [Sø97]). In [Gir87b] (p.150-159), the author adapted Gandy's proof for Gödel's system T (see [Gan80]) to ND . The general method proposed by Gandy can be applied to any logical system S as follows:

- modify S in such a way that "nothing is lost" during normalization⁴ (rewrit-

² Things went actually the other way round: cut-elimination was proven by Gentzen *in order* to prove consistency of Peano Arithmetic PA .

³ A term associated with a proof in a logical system is often said to be *typed*, the conclusion of the proof is a *type* of the term.

⁴ Thanks to the Curry-Howard correspondence one can use the language of λ -terms for proofs, and vice versa.

ing steps never erase pieces of proofs): let’s call $\rightsquigarrow_{\neg e}$ the rewriting rule of S “without erasing steps”,

- prove WN for $\rightsquigarrow_{\neg e}$ (we ’ll sometimes write in the sequel that S enjoys $WN^{\neg e}$) and confluence⁵ for $\rightsquigarrow_{\neg e}$,
- define on the proofs of S an increasing size w.r.t. $\rightsquigarrow_{\neg e}$,
- conclude that S enjoys SN.

Notice that this method “computes” an upper bound for the length of the $\rightsquigarrow_{\neg e}$ reduction sequences starting from any proof π of S : the size of π ’s normal form w.r.t. $\rightsquigarrow_{\neg e}$.

Linear Logic (LL [Gir87a]) is a refinement of intuitionistic logic and of classical logic, which gives a *logical* status to the operations of erasing and copying (corresponding to the *structural rules* of intuitionistic logic and of classical logic). This change of viewpoint on structural operations has striking consequences: one of the most important is the introduction of proof-nets (simply called nets in this paper), a geometric way of representing computations, where the strict distinction between inputs and outputs completely vanishes (this is a very strong difference between nets and λ -terms). In LL, cut-elimination is defined directly for proof-structures (general graphs which are not necessarily proofs), and nets are the “correct” proof-structures (the ones satisfying a *correctness* condition). The presence of proof-structures as new computational objects widens the space of the possible interactions between (logical) agents: this makes the system much richer, more interesting (and more complicated). We’ll show how, for the termination property of LL proof-structures, not only the distinction typed/untyped is relevant (this was already the case for λ -terms), but also the one correct/noncorrect (which is independent from the typed/untyped one).

We present in the paper the first complete proof of SN for full second order LL: we prove SN for the “modern” version of Girard’s nets (the nets of [TdF03]). Because the system contains PA , by Gödel’s theorem the proof of SN cannot be carried out within PA : it uses Girard’s reducibility candidates (introduced in [Gir72] and already used for LL in [Gir87a]). However, by applying Gandy’s method to LL, we show that strong principles are needed only to prove WN (more precisely $WN^{\neg e}$).

Some readers might be surprised that so many years after LL’s birth (and after so many papers on the subject) no complete proof of SN for LL has been given. In [Gir87a] Girard gives a “proof” of SN based on the standardization theorem (theorem 4.25 p.72 of [Gir87a]), which is not proven in the paper. Intuitively, the standardization theorem states that if a net can be transformed into a $\rightsquigarrow_{\neg e}$ -normal form “without applying erasing steps”, then it enjoys SN. This basically corresponds to achieving all the tasks of Gandy’s method, except the proof of $WN^{\neg e}$.

⁵ A rewriting system enjoys the confluence property when if $t \twoheadrightarrow t_1$ and $t \twoheadrightarrow t_2$, there always exists t' such that $t_1 \twoheadrightarrow t'$ and $t_2 \twoheadrightarrow t'$, where t, t_1, t_2, t' are terms and \twoheadrightarrow is the reflexive and transitive closure of the rewriting rule of the system.

In LL, the absence of a unique output (actually the vanishing of the distinction inputs/outputs) entails the absence of a distinguished cut (something like a “head cut”). There still exists some kind of hierarchy on cuts in LL nets (namely the so-called *exponential depth*), but no difference can be made between two cuts with the same depth. Worse, the reduction of a cut may seriously affect the status (and thus the potential reduction) of other cuts at the same exponential depth. There is no analogue of such a meddling of cuts in the life of their fellows with the same exponential depth in the λ -calculus: the β -reduction of a redex may affect other redexes only if these last ones are deeper than the former one (in LL’s terminology: the affected redexes must have a greater exponential depth than that of the reduced one). The reader can refer to Paragraph 2.4.2 (where the notion of *exponential dependence* is introduced to overcome this difficulty) for a more precise discussion. All this makes the standardization theorem an essential ingredient of the SN proof for second order LL, which *is not* a straightforward adaptation of the WN proof. This is in sharp contrast with what happens for Girard’s system F (see [Gir72]). The presence of an head redex in F’s terms makes it easy to turn the proof of WN into a proof of SN (by a slight modification of the definition of reducibility candidate): SN is an easy variant of WN. One might find more elegant to distinguish WN from SN (following Gandy), but in F it is still possible to mix them without losing control on the combinatorial part of the SN proof. This becomes very difficult in LL. In [Dan90], the “propriété de striction” (théorème 8.31 p.64) allows to complete Girard’s proof for second order multiplicative and exponential LL (MELL₂), a significant fragment of LL (containing system F): SN of several other classical and linear systems has been proven thanks to appropriate embeddings in MELL₂ (see for example [DJS97] and [LQTdF05]). But up to now, no proof of SN is available for the full system, essentially because of the presence of the additive connectives, whose computational behaviour is difficult to handle (for example, cut-elimination is not confluent in presence of the additives, at least in the traditional syntax). The main goal of the paper is to finally fill this (rather big) hole in LL’s litterature. For the sake of completeness, we mention here two previous attempts to prove this result: annexe A of [Tdf00] and [Oka99]. In annexe A of [Tdf00] a proof of a (variant of the) standardization theorem is given but (as it is explained), it is not enough to prove SN for full LL. In [Oka99], a nice approach to termination using phase semantics is proposed, which is suitable for WN (at least in the fragment MELL₂) but not for SN: on the one hand the proof of standardization for MELL₂ is not convincing (it is only “sketched” as the author writes) and on the other hand in presence of the additives the considered cut-elimination procedure is not the full one (see Subsection 4.2 for a detailed discussion on [Oka99]).

Our approach is to start from scratch, having in mind the two following guidelines:

- as soon as the system is powerful enough, the combinatorial part of the SN proof (standardization or any of its variants) should be split from the part involving strong principles (WN or any of its variants);

- it should be stressed where logic (more precisely types and correctness) comes into the picture: to which extent is it possible to compute with untyped proof-structures?

Our true main result is a standardization theorem (Theorem 3.2), proven for “sliced pure structures” (sps), which are much more general than nets.

The notion of “slice” was introduced in [Gir87a] in order to reduce the difficulty of dealing with the additive connectives. In the polarized framework, its variant “sliced proof-structure” is proven to enjoy nice properties ([LTdF04]). The main reason why we use slices in this paper is that we need to prove a confluence property (remember it is part of Gandy’s method), and it is the only known syntax for LL for which one can hope to prove such a property.

Our sps are an extension of sliced proof-structures to full *untyped* LL. Indeed, following the idea that types and correctness should be used only when necessary, we start our analysis with the most general kind of graph we have: untyped and noncorrect. We show in Section 2 how one can always compute with such general structures. This is a first novelty of our paper: both in [Gir87a] and [Tdf03], in presence of the additives, computations are defined only for nets (typed and correct proof-structures). However, without any kind of correctness, computations behave very badly (and this is not related to the presence/absence of types): we give a counterexample both to WN and to confluence for such general (noncorrect) sps (see Figure 12). We then introduce the correctness condition “acyclicity slice by slice” (AC condition, Definition 2.9). It is well-known that this condition is far too weak to characterize proofs (see [Gir96]), and it took a lot of time to find the right condition one has to add to the correctness slice by slice in order to characterize logical correctness (see [HvG03]). What we show here is that, despite its weakness, the AC condition is enough to prove confluence (Theorem 3.12) and standardization (Theorem 3.2): from the computational point of view only a very weak form of correctness is needed.

Let us now describe more precisely the structure of the paper. As the connection between the different results is rather delicate and complex, we conclude the introduction by representing the tree-like structure of the dependencies between the lemmata, propositions, theorems (see Figure 1). Also, the whole paper is scattered with examples and figures: we hope this will help the reader.

Section 2 is devoted to sps. In Subsection 2.1 we define sps (Definition 2.2) and their cut-elimination (Definition 2.6). Subsection 2.2 is entirely devoted to motivating our results and choices by several examples and counterexamples. In Subsection 2.3 we introduce the weak form of correctness used in the paper and expressed by the AC condition of Definition 2.9. Subsection 2.4 achieves a first essential step of our proof: Definition 2.11 splits the cut-elimination rewriting rule (denoted by \rightsquigarrow_{cut}) into two strongly normalizing reductions, the logical reduction (\rightsquigarrow_{log} , proven to enjoy SN in Proposition 2.14) and the structural reduction (\rightsquigarrow_{str} , proven to enjoy SN in Proposition 2.20). These two reductions are disjoint and their union is \rightsquigarrow_{cut} ; they both enjoy SN on sps satisfying AC, even if their union \rightsquigarrow_{cut} does not even

enjoy WN (the untyped λ -calculus can be embedded in sps satisfying AC, see in particular the example of Fig. 10). Propositions 2.14 and 2.20 are essential ingredients in the proof of confluence (Theorem 3.12) of next section.

In Section 3 we apply Gandy’s method to sps: we distinguish the erasing cuts from the non erasing ones and we define the $\rightsquigarrow_{\neg e}$ rewriting rule for sps (Definition 3.1). We then explain (Subsection 3.1) why SN of $\rightsquigarrow_{\neg e}$ entails SN of \rightsquigarrow_{cut} : a postponement lemma (Lemma 3.3) allows to “delay” erasing steps after any non-erasing one, from which the result easily follows (Proposition 3.5). Subsection 3.2 shows that from WN of $\rightsquigarrow_{\neg e}$ one can deduce SN of $\rightsquigarrow_{\neg e}$: the key point here is the confluence theorem (Theorem 3.12), proven for a *labelled* version of sps (see Definitions 3.6 and 3.7). We first prove that labels allow to define an increasing size on (labelled) sps⁶, and that (assuming confluence holds) this allows to prove $WN^{\neg e} = SN^{\neg e}$ for sps satisfying AC: from this equality and Subsection 3.1 our main result (the standardization, Theorem 3.2) immediately follows. The rest of Section 3 is devoted to the (delicate) proof of confluence, based on the following results:

- (1) confluence of the labelled version of the logical reduction \rightsquigarrow_{log} of Subsection 2.4: it follows from local confluence (Lemma 3.14) and SN (immediate consequence of Proposition 2.14 of Subsection 2.4)
- (2) confluence of the labelled version of the structural reduction \rightsquigarrow_{str} of Subsection 2.4: it follows from local confluence (Lemma 3.16) and SN (immediate consequence of Proposition 2.20 of Subsection 2.4)
- (3) commutation of \rightsquigarrow_{log} and \rightsquigarrow_{str} (Lemma 3.19)
- (4) Hindley-Rosen lemma (see [Klo92], [Py98]): a rewriting rule which is the union of two confluent rewriting rules which commute is itself confluent.

It is important to stress the fact that confluence (so as the main result of the paper Theorem 3.2) is far from being an immediate consequence of the same result for MELL. It is only partially true that sps allow to work “slice by slice”: the additive commutative normalization step (the nightmare of normalization in presence of the additives, see Section 4 and Figure 19) is not explicitly present in our syntax, but it is hidden in other normalization steps (the $(!/?d)$ and the $(!/!)$ steps of Definition 2.6). This is highly moral, since the exponential connectives are the bridge between the additive and the multiplicative worlds through the isomorphism $!(A \& B) = !A \otimes !B$.

The last Section 4 is devoted to prove $WN^{\neg e}$ of \rightsquigarrow_{cut} for (second order LL) nets. As already mentioned, the AC condition is not enough to prove $WN^{\neg e}$ for sps, since AC sps contain the (untyped) λ -calculus: we already know that there is no hope to prove WN (nor SN) without some constraints. One could actually stop at the end of Section 3 and say that any (reasonable version of second order LL) net is an AC sps which satisfies $WN^{\neg e}$ (proof: adapt Girard’s reducibility candidates method). But in order to obtain a precise and complete result, we consider the nets of [Tdf03] with units: this syntax is described in Subsection 4.1. The next Subsection 4.2 con-

⁶ Notice that the AC condition is not needed here.

tains the motivations for the strategy we use to prove SN of LL (Theorem 4.10), and relates our results to previous attempts (namely [Gir87a] and [Oka99]). In Subsection 4.3, we notice that it is very easy to associate with every net β its “slicing” $sl(\beta)$ which is an sps, and that the cut-elimination of β can be simulated by that of $sl(\beta)$ (Lemma 4.3 and Figure 23). Finally, in Subsection 4.4 we conclude thanks to the following facts: (i) $sl(\beta)$ satisfies AC (Proposition 4.1), (ii) if $sl(\beta)$ enjoys SN then β enjoys SN (Proposition 4.2), (iii) $sl(\beta)$ enjoys WN^{-e} (Theorem 4.9). Standardization for sps (Theorem 3.2) allows then to conclude SN for nets (Theorem 4.10).

Contents

1	Introduction	1
2	Sliced pure structures	9
2.1	Definitions	9
2.2	Examples of reductions	18
2.3	The AC property	22
2.4	Two results of strong normalization	22
3	Standardization for sliced pure structures	36
3.1	SN is a consequence of SN^{-e}	36
3.2	SN^{-e} is a consequence of WN^{-e}	38
4	Strong Normalization for Linear Logic	48
4.1	The syntax of nets	48
4.2	A digression on standardization	50
4.3	Slicing nets	53
4.4	Strong normalization for nets	55
	References	59

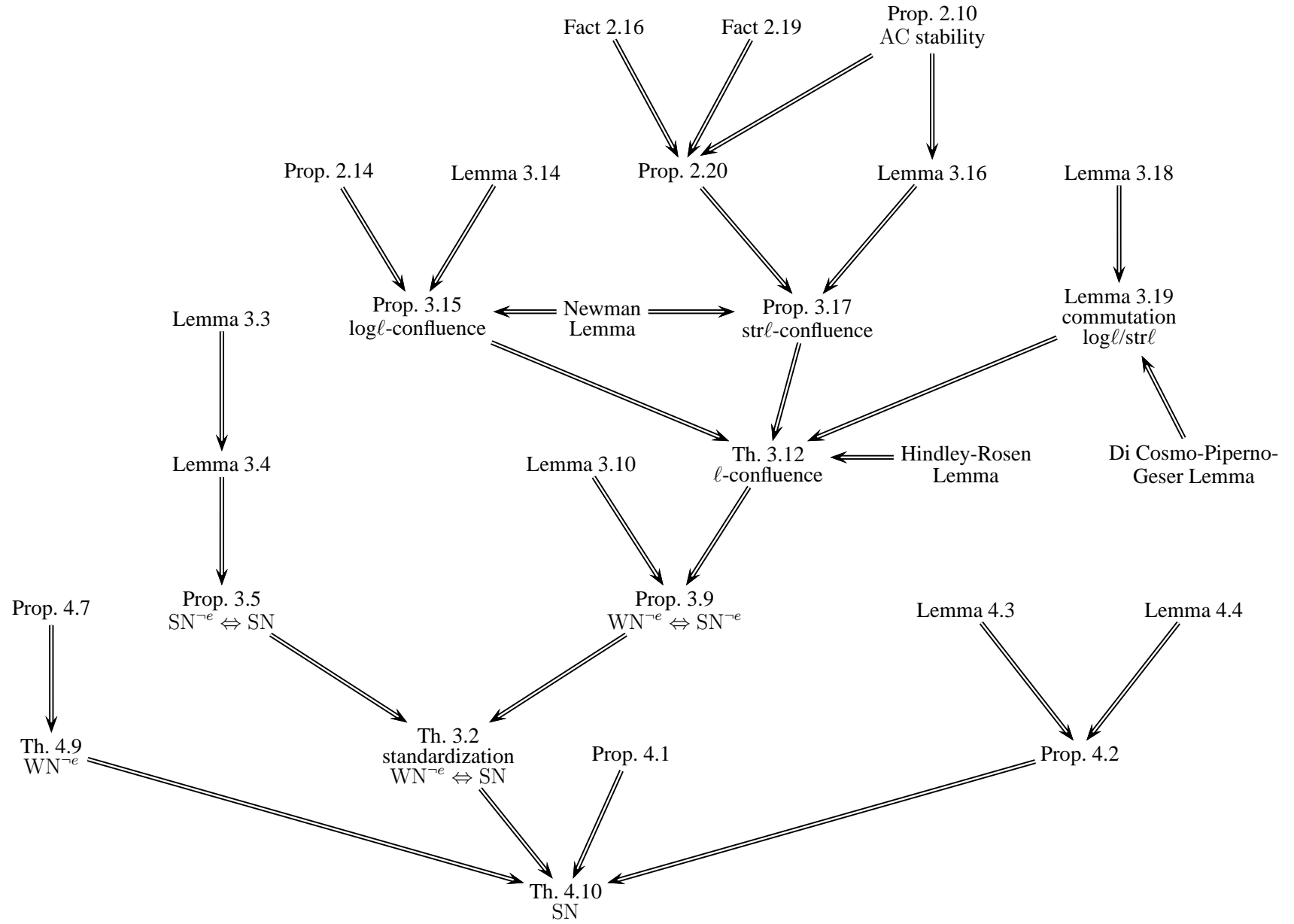


Fig. 1. Dependencies between the results of this paper

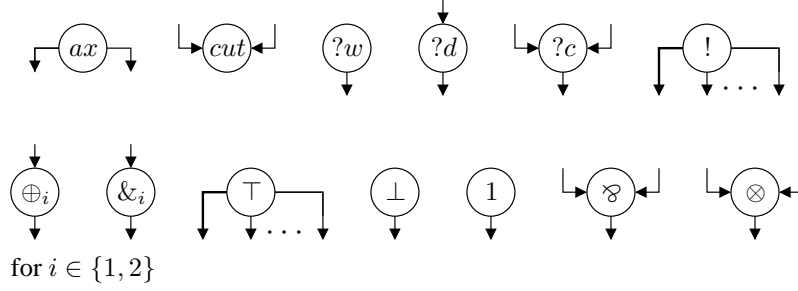


Fig. 2. Sps links

2 Sliced pure structures

In this section we define and study sliced pure structures (sps) and their reduction. After the main definitions (Subsection 2.1), we illustrate by means of several examples the notions previously introduced (Subsection 2.2); we then present the *AC* condition (Subsection 2.3) and we conclude (Subsection 2.4) by proving two strong normalization results which will be essential for the confluence theorem (Theorem 3.12) of next section 3: Proposition 2.14 and Propostion 2.20.

2.1 Definitions

We start by extending to full LL the definition of “sliced proof-structure” given in [LTdF04] for the polarized fragment. In the style of [LTdF06], we work in an untyped framework.

Definition 2.1 (Flat) *A flat is a finite (possibly empty) labelled directed acyclic graph whose nodes (also called links) are defined together with an arity and a coarity, that is a given number of incident edges called the premises of the node and a given number of emergent edges called the conclusions of the node. The valid nodes are in Figure 2.*

The !-links and the \top -links have a distinguished conclusion (denoted in Figure 2 by a bold arrow) called main conclusion of the link; the other conclusions are the auxiliary conclusions. We allow edges with a source but no target, they are called conclusions of the flat.

Links will be denoted by Latin letters l, m, \dots . Flats will be denoted by initial Greek letters α, β, \dots

When drawing a flat we represent edges oriented up-down so that we speak of moving upwardly or downwardly in the graph, and of nodes or edges “above” or “under” a given node/edge. In the sequel we will not write explicitly the orientation

of the edges.

Definition 2.2 (Sliced pure structure) A slice of depth 0 is a flat without $!$ -links. A sliced pure structure (*sps for short*) of depth 0 is a finite (possibly empty) multiset of slices of depth 0 with the same conclusions⁷.

A slice of depth $d + 1$ is a flat α such that with every $!$ -link o of α with $n_o + 1$ conclusions is associated an *sps* of depth at most d , called the *box* of o , with n_o auxiliary conclusions corresponding to the n_o auxiliary conclusions of o and another conclusion (the *main conclusion* of the box) corresponding to the main conclusion of o . Moreover α has at least one $!$ -link with a box of depth d .

A sliced pure structure (*sps for short*) of depth $d + 1$ is a finite multiset of slices of depth at most $d + 1$ with the same conclusions, and s.t. at least one of these slices has depth $d + 1$.

We denote slices by initial Greek letters α, β, \dots , *sps* by final Greek letters π, σ, \dots

The depth of a link l in an *sps* π , denoted $\text{depth}(l)$, is the number of boxes of π containing l . The size of π , denoted $\text{size}(\pi)$, is the number of links (at any depth) of π .

The reader should notice that our *sps* are *multisets* of slices, and not simply sets, as it is in [LTdF04], [LTdF06]: this quibble is needed to avoid an unnatural erasing of slices during cut-elimination. We refer to Subsection 2.2 for examples of this phenomenon. In the sequel we will speak of a slice α of π meaning an *occurrence* of the slice α in π . As a consequence, when we write $\alpha \in \pi$, we are considering an occurrence of α in the multiset π , and when that expression bounds an operator, as for example in $\sum_{\alpha \in \pi}$, we mean that $\sum_{\alpha \in \pi}$ varies on the set of occurrences of π 's slices.

Figure 3 is an example of *sps* of depth 1; the correspondence between the conclusions of the box of a $!$ -link and the conclusions of the $!$ -link is given in the figure by the order of the edges (from left to right).

Remark 2.3

- Notice that, by definition, the boxes of an *sps* satisfy a nesting condition: two boxes are either disjoint or contained one in the other.
- Once the decision to work without types has been taken, the question arising is: to which extent? A possibility was to use recursive types (like in [Dan90], [Reg92]), another one to type only $?$ -edges (like in [LTdF06]). In the general LL

⁷ Notice that in our (strongly untyped) framework, this simply means that two slices have the same number of conclusions, and that an *sps* establishes a bijection between the conclusions of every two different slices.

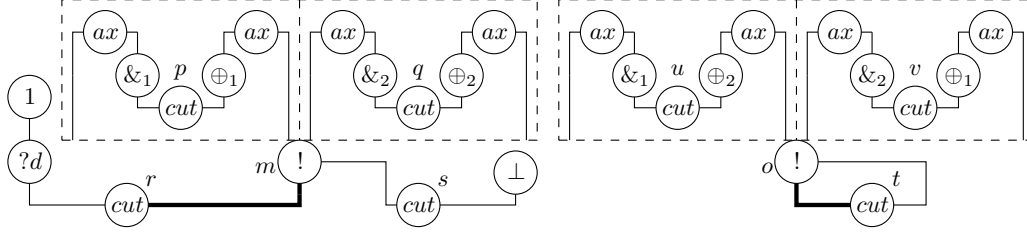


Fig. 3. An example of sps

case that we consider here, both the answers are not completely satisfactory and we decided to work in a strongly untyped framework. There are little surprises following this choice: the main one is that a clash (Definition 2.5) might become reducible after some cut-elimination steps (Definition 2.6): for example, the cut s of Figure 3 is a clash, but it becomes reducible after the reduction of the cut r (see Subsection 2.2). However such oddities cause no problem w.r.t. our purposes.

- In order to have a very general result we had to use all the links of [Gir87a], including \top . A possible (and rather natural) choice was to rule out that link and to represent the \top rule as empty sps (like in [Lau02]). Indeed, such a choice would realize a greater quotient on proofs: this notion of sps is “closer” to denotational semantics. However, the absence of \top in sps makes it more difficult to infer strong normalization of LL nets from strong normalization of sps (Proposition 4.2). More precisely, we would lose Lemma 4.3: this is the reason for our choice.

Of course all the nice properties we prove for our sps still hold for \top -free sps (in particular Theorem 3.2).

Remark 2.4 Concerning the presence of empty structures, notice that:

- the empty flat does exist (and so do the empty slice and sps containing the empty slice), and it has no conclusion. Its presence is required by the cut-elimination procedure (Definition 2.6): the procedure applied (for example) to the sps containing a unique flat consisting of a \top -link a cut and a \perp -link yields the empty graph;
- with a $!$ -link o of an sps, it is never possible to associate an sps containing the empty slice: o has at least one conclusion and this has also to be the case for the sps associated with o ;
- empty sps (that is empty multisets of slices) do exist: there is one such sps for every set of conclusions.

Conventions. Given a link l of an sps π , we will often speak of “the flat of l ” always meaning the biggest flat of π containing l .

We will sometimes refer to “the maximal flats of an sps”, meaning the flats which are not (strict) subgraphs of other flats of the sps.

As mentioned in the introduction, our sps are very general structures: they are untyped and they may be incorrect (w.r.t. the proofs of LL). Even in such a general setting cut-elimination can be defined and nice properties hold, as we will show in the sequel of the paper. However, one has to handle carefully some strange phenomena related to this rather “savage” situation, like the presence of cuts which are not reducible:

Definition 2.5 (Clash and deadlock) *The two edges premises of a cut link are dual when:*

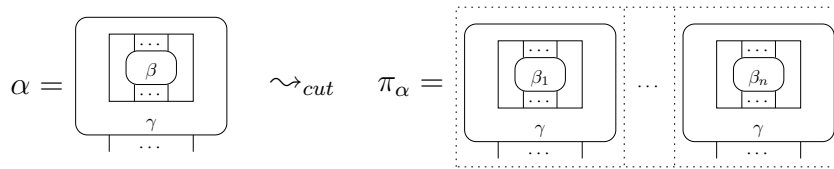
- they are conclusions of a \otimes -node and of a \wp -node,
- they are conclusions of a \oplus_i -node and of a $\&_j$ -node (for $i, j \in \{1, 2\}$),
- they are conclusions of a 1 -node and of a \perp -node,
- one is the main conclusion of a $!$ -node and the other one is either an auxiliary conclusion of a $!$ -node, or the conclusion of one of the following nodes: $?c$, $?w$, $?d$.

A cut node of an sps is:

- a clash, when the premises of the cut node are not dual edges and none of the two is the conclusion of an ax -link nor an auxiliary conclusion of a \top -link;
- a deadlock, when the two premises of the cut link are conclusions of the same ax -link (resp. $!$ -link, \top -link);
- reducible, otherwise.

In Figure 3, for example, t is a deadlock, s is a clash and the five other cuts are reducible.

We now define the cut-elimination \rightsquigarrow_{cut} on sps. Let us give an idea of how \rightsquigarrow_{cut} works on a slice α (the extension to general sps will be immediate). To eliminate a cut in α means in general to transform α in an sps $[\alpha'_i]_{i \in I}$, s.t. every slice α'_i is obtained from α by substituting a specific subgraph β of α with a subgraph β_i having the same pending edges (i.e. edges with no target or no source) as β , as pictured below:



The number n and the subgraphs $\beta, \beta_1, \dots, \beta_n$ depend on the cut we want to reduce (see Fig. 4). In case $n = 1$ then π_α is a singleton, that is a slice (this will be the case for every type of cut but the types $(\&_i/\oplus_j)$ and $(!/?d)$); in case $n = 0$, then π_α is the empty multiset of slices.

Definition 2.6 (Types of cut and cut-elimination) *Let π be an sps, let t be a reducible cut link of π , and let α be the (biggest) slice in π containing t at depth 0.⁸ We define the multiset of slices $\pi_\alpha = [\alpha'_i]_{i \in I}$, obtained by applying some transformations to α (these depend on the type of the cut t). The one step reduct $\bar{\pi}$ of π is obtained from π by substituting π_α for α . All cases are pictured in Figure 4:*

- *(ax): one premise of t is the conclusion of an ax-link and the other one is not the auxiliary conclusion of a \top -link. In this case $[\alpha']$ is obtained from α as usual (i.e. by erasing the axiom link, its conclusions and the cut link, and by connecting what has to be connected);*
- *(\otimes/\wp): one premise of t is the conclusion of a \otimes -link, the other one is the conclusion of a \wp -link. In this case $[\alpha']$ is obtained from α by erasing the \wp -link, the \otimes -link and the cut link t (and its premises) and by putting two new cut links between the two left (resp. right) premises of the \wp -link and of the \otimes -link⁹;*
- *($1/\perp$): one premise of t is the conclusion of a 1-link, the other one is the conclusion of a \perp -link. In this case $[\alpha']$ is obtained from α by erasing the three links: 1, \perp and the cut (and its premises);¹⁰*
- *($\oplus_i/\&_j$): one premise of t is the conclusion of a \oplus_i -link, the other one is the conclusion of a $\&_j$ -link. If $i = j$, then $[\alpha']$ is obtained from α by erasing the two links (and their conclusions) and by moving up the cut link to their premises. If $i \neq j$, then $I = \emptyset$ (we simply erase α);*
- *($!/?d$): one premise of t is the main conclusion of a $!$ -link and the other one is the conclusion of a $?d$ -link. Let τ be the sps associated with the $!$ -link. If $\tau = \emptyset$, we erase α (i.e. $I = \emptyset$). Otherwise, with each slice β_i of τ , we associate the slice α'_i defined by erasing the cut link, the $?d$ -link (and its conclusion), by replacing in α the $!$ -link (and its conclusions) by β_i , and by cutting β_i 's main conclusion¹¹ with the premise of the $?d$ -link;*
- *($!/?w$): one premise of t is the main conclusion of a $!$ -link and the other one is the conclusion of a $?w$ -link. In this case, the $!$ -link (together with its box) and its conclusion edges are erased. We then erase the $?w$ -link, the cut and its premises, and we add as many $?w$ -links as the auxiliary conclusions of the $!$ -link, thus obtaining the unique slice $[\alpha']$;*
- *($!/?c$): one premise of t is the main conclusion of a $!$ -link, and the other one is the conclusion of a $?c$ -link. In this case, $[\alpha']$ is obtained from α as follows: let's call l the $!$ -link, n the $?c$ -link and a_1 and a_2 the two edges premises of n . We create a new $!$ -link l' by copying the link l , and we pairwise contract the auxiliary conclusions of l and l' : the conclusions of these new $?c$ -links substitute the auxiliary conclusions of l in α . We then erase n (and its conclusion) and t ,*

⁸ One can also assume that t has depth 0 in π : the generalization is straightforward.

⁹ Notice that this means that the premises of the \otimes/\wp -links are *ordered*; we shall see in the transformation associated with the $(!/?c)$ cut link that this is not the case of the premises of the $?c$ -links (nor of the premises of the cut links).

¹⁰ This case -so as the $(!/?w)$ one- might yield an empty graph.

¹¹ We extend here the notion of “main conclusion of a box” to every slice of the box.

and we connect the main conclusion of l (resp. l') with a_1 (resp. a_2) by means of a cut link. The sps associated with l and l' are the same;

- $(!/!)$: one premise of t is the main conclusion of a $!$ -link l , the other one is an auxiliary conclusion of a $!$ -link l' . If we call a' the auxiliary conclusion of l' premise of t and τ the sps associated with l' , then $[\alpha']$ is obtained from α by erasing t and l and its conclusions, and by replacing the conclusion a' of l' by all the auxiliary conclusions of l . If $\tau = \emptyset$, with this new $!$ -link (which we still denote by l') is associated the empty sps. If $\tau \neq \emptyset$, with l' is associated the sps obtained by substituting every slice β of τ by the slice obtained by cutting the auxiliary β 's conclusion corresponding to a' with the main conclusion of l (the sps associated with l remains unchanged);
- $(\top - cc)$: one premise of t is an auxiliary conclusion of a \top -link l ; let's call a this edge. In this case $[\alpha']$ is obtained from α as follows: we (arbitrarily) select a slice β (not containing l) having the other premise of t among its conclusions. We then substitute the slice composed of l , the cut link t and the slice β by a new \top -link (which we still call l), having the same conclusions as the original l where we have substituted the edge a by the conclusions of β (different from t 's premise).

We denote by $t(\pi)$ the sps¹² obtained applying the transformation previously defined associated with the cut link t . We'll also refer to $t(\pi)$ as a one step reduct of π , and to the transformations associated with the different types of cut link as the reduction steps.

In the sequel we will denote the set WN^{cut} (resp. SN^{cut}) of weakly (resp. strongly) normalizable sps w.r.t. \rightsquigarrow_{cut} simply by WN (resp. SN).

Remark 2.7

- Notice that the cut-elimination procedure is defined without any reference to correctness.
- The reader certainly noticed the restriction imposed in the (ax) case: in order for a cut to be of type (ax) , not only one premise of the t must be conclusion of an axiom, but also the other premise cannot be an auxiliary conclusion of a \top -link. This restriction makes every cut link of an sps of a unique type: in the absence of it, there would be a (unique) case in which a reducible cut link t of an sps might have two different types ((ax) and $(\top - cc)$). Notice that this little problem would not occur if \top were rejected from sps links (see Remark 2.3). Anyway, this is actually no restriction at all, because there is a(n obvious) $(\top - cc)$ reduction step having exactly the same effect as the (ax) reduction step: erase the axiom link. Working with reducible cut links having a unique type is (simpler and) useful in Section 3, when we define the notion of erasing cut (Definition 3.1).
- Notice that the $(\top - cc)$ step gives rise to non deterministic (and non conflu-

¹² The fact that $t(\pi)$ is indeed an sps can be easily checked.

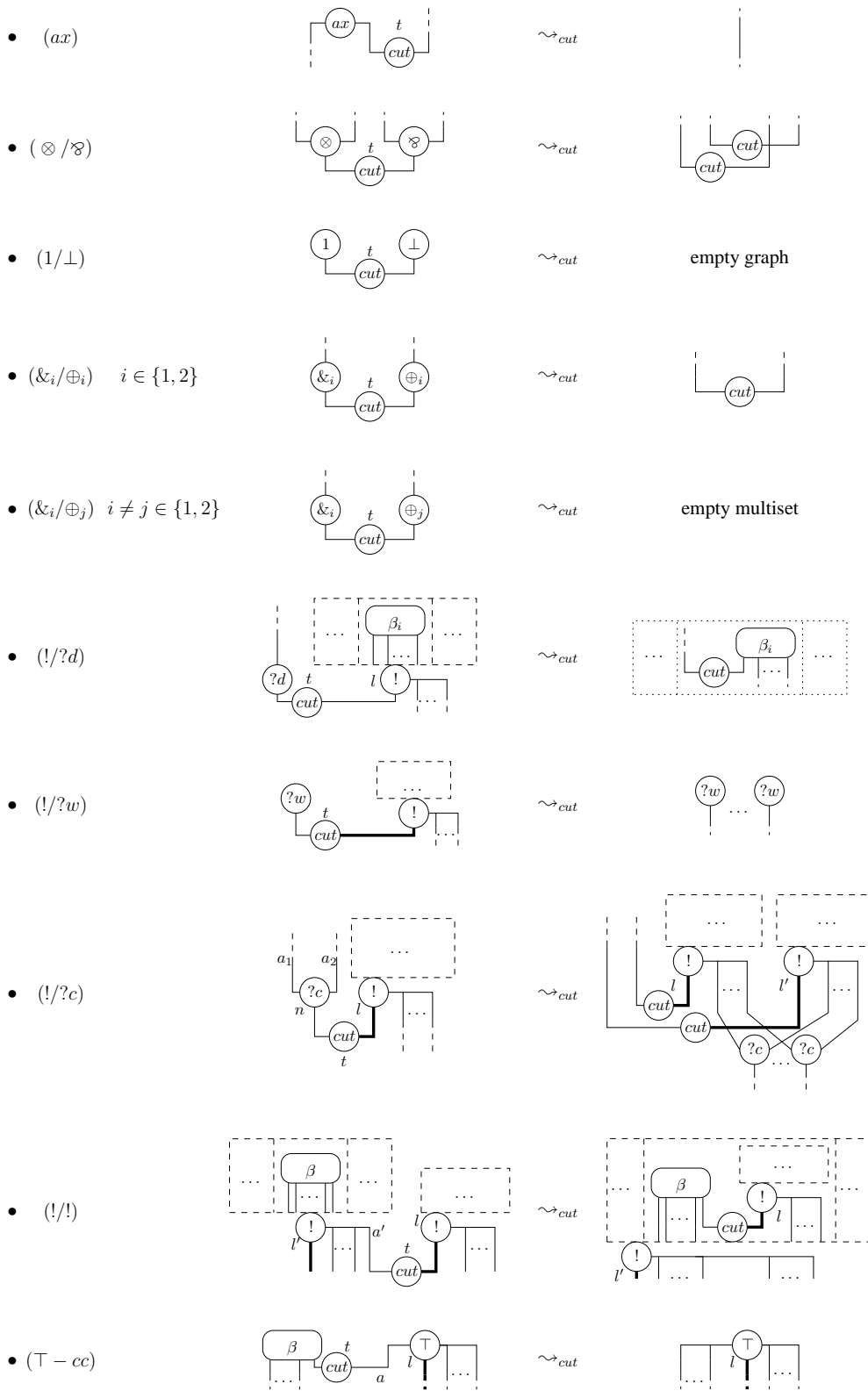


Fig. 4. Cut-elimination for sps

ent) reductions: for example, a cut whose premises are auxiliary conclusions of two distinct \top -links l, l' can be reduced by erasing either l or l' . Anyway such phenomena disappear when one considers only non erasing reductions (see Def. 3.1).

We now give a precise definition of the notions of ancestor and residue of a node: the point is to know whether a node l of $t(\pi)$ is “created” by the cut-elimination procedure or “residue” of some π ’s node.

Definition 2.8 (Ancestor, residue) *Let π be an sps, t be a cut link of π and $t(\pi)$ be a one step reduct of π associated with t . When a node l of $t(\pi)$ comes from a (unique) node \overleftarrow{l} of π , we say that \overleftarrow{l} is the ancestor of l in π and that l is a residue of \overleftarrow{l} in $t(\pi)$. If this is not the case, then l has no ancestor in π , and we say it is a created node. We indicate, for every type of cut node t of Definition 2.6, which links are created in $t(\pi)$ (meaning that the other nodes of $t(\pi)$ are residues of some π ’s node). We use the notations of Definition 2.6 and Figure 4:*

- (ax) : there are no created nodes in $t(\pi)$;
- (\otimes/\wp) : the two new cut links between the two left (resp. right) premises of the \wp -link and of the \otimes -link are created nodes;
- $(1/\perp)$: there are no created nodes in $t(\pi)$;
- $(\oplus_i/\&_j)$: if $i = j$, then the cut link between the two premises of the $\oplus_i/\&_i$ -links is a created node. If $i \neq j$, there are no created nodes in $t(\pi)$;
- $(!/?d)$: every cut link between β_i ’s main conclusion and the premise of the $?d$ -link is a created node;
- $(!/?w)$: all the $?w$ -links added during this step are created links;
- $(!/?c)$: the new $?c$ -links having as premises the auxiliary conclusions of l and l' are created nodes. The two cut links having among their premises the main conclusions of l and l' are created nodes;¹³
- $(!/!)$: every cut link between the auxiliary β ’s conclusion corresponding to a' and l ’s main conclusion is a created link. (Notice that the “new” $!$ -link l' is considered a residue of the corresponding $!$ -link of π , even though it might have different conclusions);
- $(\top - cc)$: there are no created nodes in $t(\pi)$.

We conclude this subsection with a short recall of some standard terminology in rewriting theory, which will be used in the sequel (see [Klo92]; we refer also to the clear and simple presentation in [Py98]).

¹³ Notice that l and l' are both residues of l .

Let \rightsquigarrow_x be a reduction on sps, then:

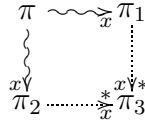
- $\rightsquigarrow_x^=$ denotes the reflexive closure of \rightsquigarrow_x ,
- \rightsquigarrow_x^+ denotes the transitive closure of \rightsquigarrow_x ,
- \rightsquigarrow_x^* denotes the reflexive and transitive closure of \rightsquigarrow_x .

Normal form: a \rightsquigarrow_x -normal form is an sps π s.t. there is no sps π_1 with $\pi \rightsquigarrow_x \pi_1$.

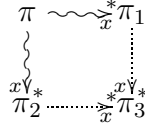
Weak normalization: an sps π is *weakly \rightsquigarrow_x -normalizable* whenever there is a normal form π_1 s.t. $\pi \rightsquigarrow_x^* \pi_1$. We denote by WN^x the set of weakly \rightsquigarrow_x -normalizable sps.

Strong normalization: an sps π is *strongly \rightsquigarrow_x -normalizable* whenever every sequence of \rightsquigarrow_x -steps starting from π is finite. We denote by SN^x the set of strongly \rightsquigarrow_x -normalizable sps.

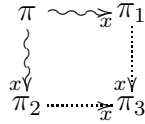
Local confluence: \rightsquigarrow_x is *locally confluent* if for every π, π_1, π_2 s.t. $\pi \rightsquigarrow_x \pi_1$ and $\pi \rightsquigarrow_x \pi_2$, there is π_3 s.t. $\pi_1 \rightsquigarrow_x^* \pi_3$ and $\pi_2 \rightsquigarrow_x^* \pi_3$. We sketch this property with the following diagram:



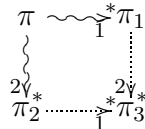
Confluence: \rightsquigarrow_x is *confluent* if for every π, π_1, π_2 s.t. $\pi \rightsquigarrow_x^* \pi_1$ and $\pi \rightsquigarrow_x^* \pi_2$, there is π_3 s.t. $\pi_1 \rightsquigarrow_x^* \pi_3$ and $\pi_2 \rightsquigarrow_x^* \pi_3$. We sketch this property with the following diagram:



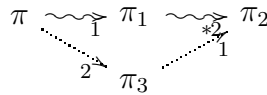
Strong confluence: \rightsquigarrow_x is *strongly confluent* if for every π, π_1, π_2 s.t. $\pi \rightsquigarrow_x \pi_1$ and $\pi \rightsquigarrow_x \pi_2$, there is π_3 s.t. $\pi_1 \rightsquigarrow_x \pi_3$ and $\pi_2 \rightsquigarrow_x \pi_3$. We sketch this property with the following diagram:



Commutation: two reductions \rightsquigarrow_1 and \rightsquigarrow_2 *commute*, if for every π, π_1, π_2 s.t. $\pi \rightsquigarrow_1^* \pi_1$ and $\pi \rightsquigarrow_2^* \pi_2$, there is π_3 s.t. $\pi_1 \rightsquigarrow_2^* \pi_3$ and $\pi_2 \rightsquigarrow_1^* \pi_3$. We sketch this property with the following diagram:



Postponement: the reduction \rightsquigarrow_1 can be postponed w.r.t. the reduction \rightsquigarrow_2 , when for every π, π_1, π_2 s.t. $\pi \rightsquigarrow_1 \pi_1$ and $\pi_1 \rightsquigarrow_2 \pi_2$, there is π_3 s.t. $\pi \rightsquigarrow_2 \pi_3$ and $\pi_3 \rightsquigarrow_1^* \pi_2$. We sketch this property with the following diagram:



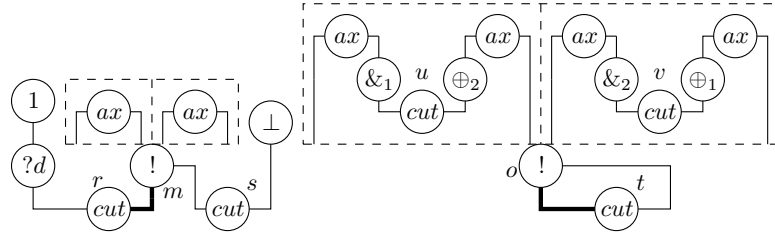


Fig. 5. The result of applying cut-elimination to the sps of Figure 3

2.2 Examples of reductions

After so many definitions, some examples might be useful. . . . Let us apply cut-elimination to the sps π of Figure 3. If you reduce the $(\&x_1/\oplus_1)$ cut p , the $(\&x_2/\oplus_2)$ cut q , and then the two created cuts of type ax , you obtain the sps π' of Figure 5. Notice that the sps π^m associated with the $!$ -link m has now two occurrences of the same slice (consisting of an ax -link): if we had defined the sps as *sets* of slices, we would have missed one occurrence of the slice in π^m , so giving an “erasing” feature to the $(\&x_i/\oplus_i)$ step which is quite unnatural. As already mentioned in the Introduction, it is crucial in order to apply Gandy’s method to have a good notion of *erasing* cut-elimination step: this will appear clear in Section 3, where we split cut-elimination (the \rightsquigarrow_{cut} rewriting rule) in the erasing and the non-erasing reduction (see Definition 3.1).

Let us go on in the elimination of the cuts in the sps π' of Figure 5. If you reduce the $(!/?d)$ cut r , then you get the sps π'' of Figure 6. Notice that this reduction duplicates the unique slice of π' , since it opens a box containing two slices. This is a tricky feature of sps (due to the presence of additive slices): the reduction of a cut of type $(!/?d)$ (like r) may duplicate other cuts at the same exponential depth (like the cuts t and s in π'). We tame this kind of duplication in Subsection 2.4, where we prove that the “logical” subreduction of \rightsquigarrow_{cut} (Definition 2.11) is strongly normalizing (Proposition 2.14).

Yet another remark on the reduction of r : notice that the residues of the clash s of Figure 5 are reducible cuts of Figure 6. Thus, let us reduce these residues of s ,

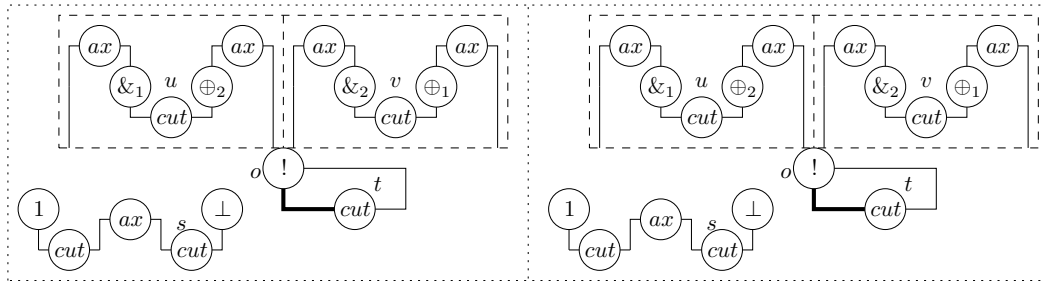


Fig. 6. The result of applying cut-elimination to the sps of Figure 5

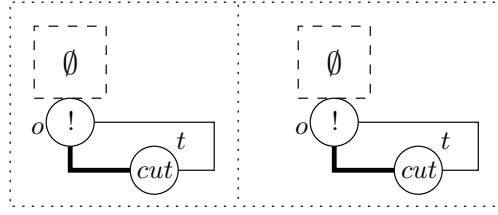


Fig. 7. The result of applying cut-elimination to the sps of Figure 6

the $(1/\perp)$ cuts so obtained and the two $(\&_i/\oplus_j)$ ($i \neq j$) cuts u and v in the box associated with o : we obtain the sps π''' of Figure 7. Notice that the two reduction steps of type $(\&_i/\oplus_j)$ ($i \neq j$) have erased both the slices of the sps associated with o , transforming it in the empty multiset of slices. This is really different from the reduction of the cut of type $(1/\perp)$, which has transformed the subgraph consisting in the 1-link, the \perp -link and the cut in the empty graph: pay attention not to confuse empty sps (i.e. empty multiset of slices) with the empty slice (see also Remark 2.4). The sps π''' of Figure 7 is a \sim_{cut} -normal form. Notice however that π''' contains the deadlock t : in general \sim_{cut} -normal forms may contain non-reducible cuts, i.e. clashes or deadlocks.

Let us come now to the problem of normalization. The cut-elimination procedure applied to an sps π may lead to infinite reduction sequences basically in two distinct cases: (i) either because π is untypable (by LL formulas, see the grammar in Figure 17) or (ii) because π is not correct (w.r.t. a notion of correctness which will be introduced in the following Subsection 2.3). We give an example of (i) in Figure 9, and examples of (ii) are in Figure 11 and in Figure 12. Let us comment a bit each of them.

The sps $\delta\delta$ of Figure 9 is taken from [Oka99]. Morally $\delta\delta$ is a net version of the most famous λ -term which is not normalizable: $(\Delta)\Delta$, where $\Delta = \lambda x.(x)x$. In Figure 10 you see a proof that $\delta\delta$ reduces to itself. The sps $\delta\delta$ is not even weakly

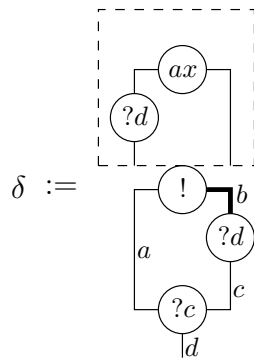


Fig. 8. The slice δ .

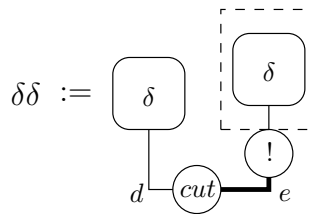


Fig. 9. The slice $\delta\delta$ which is not WN (δ is the slice of Figure 8).

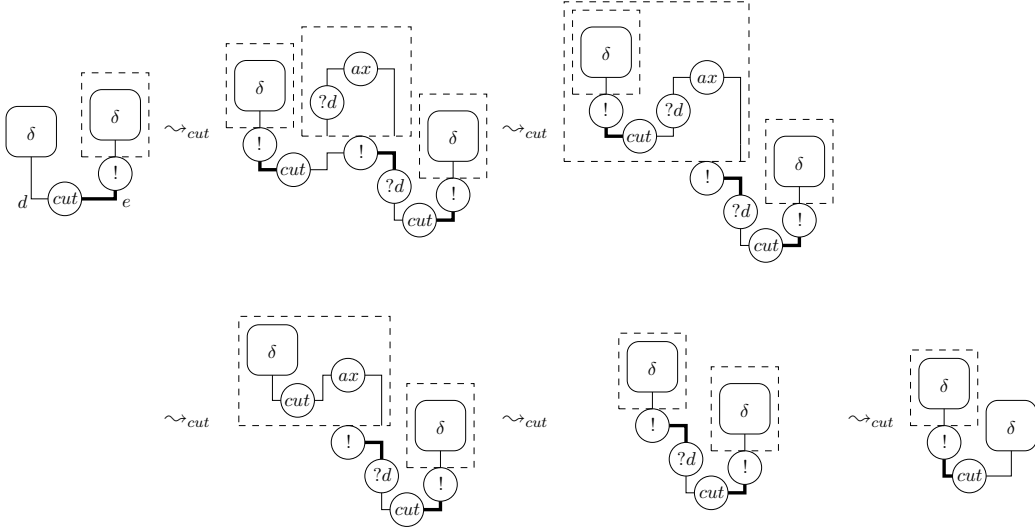


Fig. 10. A proof that $\delta\delta \rightsquigarrow_{cut}^+ \delta\delta$

normalizable, and it is not typable by LL formulas, even if it is correct in the sense that it satisfies the AC condition of Definition 2.9.

Let us now consider the slice α of Figure 11: α is typable by LL formulas, but it is not switching acyclic (Definition 2.9) owing to the cycle crossing the cut, the $!$ -link and the $?c$ -link. Morally such a cycle causes the loop pictured in Figure 11.

We can use the slice α of Figure 11 to show a last intriguing example of cut-elimination. Let γ be the (maximal) subslice of α which does not contain the cut link, then consider the slice β defined in Figure 12: β is a counter-example both to the confluence and to the normalization of cut-elimination for sps which are not correct (they don't satisfy the AC condition of Definition 2.9). On the one hand, if you reduce the $(!/?d)$ cut t , then the created cut $(!/?d)$ and last the created (\otimes/\wp) cut, you obtain the slice pictured on the right of β , which is not weakly normalizable since reducing the cut t' leads to a looping cut-elimination, similar to the one described in Figure 11. On the other hand, if you reduce the cut u in β , then you obtain the slice pictured below β , which is even strongly normalizable: its (unique)

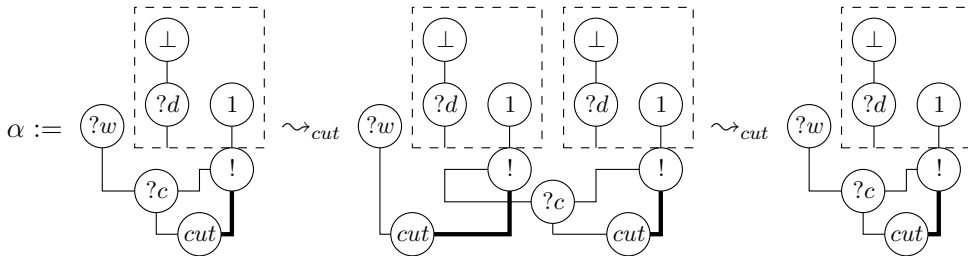


Fig. 11. An example of slice α s.t. $\alpha \rightsquigarrow_{cut}^+ \alpha$. The slice α is typable by LL formulas

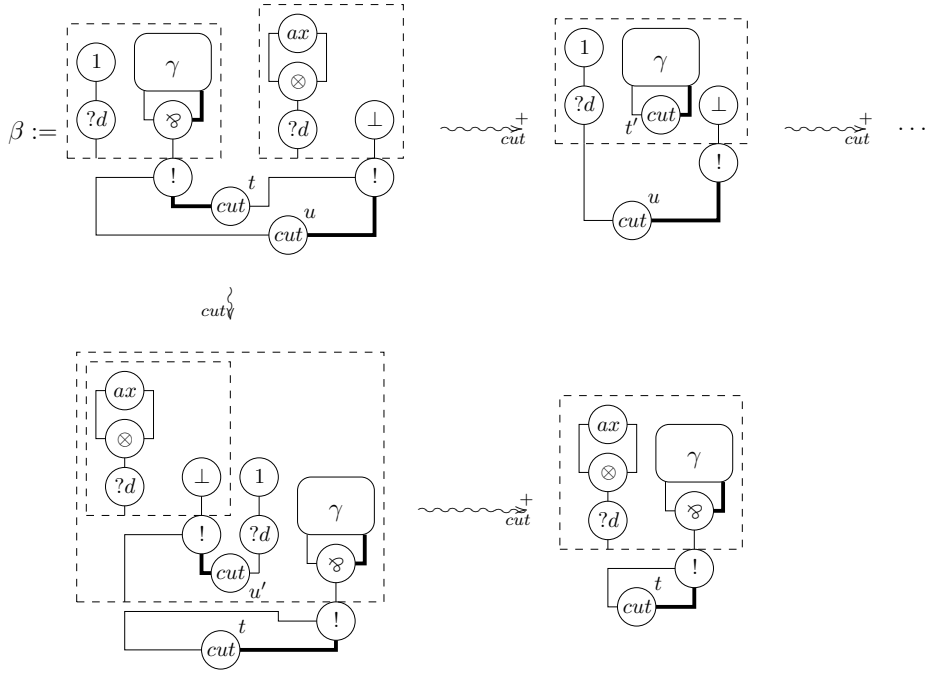


Fig. 12. A counter-example to normalization and confluence of cut-elimination. The slice γ is obtained by removing the cut from the slice α pictured in Figure 11

normal form is given on its right (notice that the cut t has become a deadlock, so it is not reducible any more). This example clearly shows that if we drop the AC condition, we lose both confluence and weak (thus strong) normalization *even for typable sps*.

The main result of this paper is the standardization theorem for correct sps (i.e. for sps satisfying AC, Definition 2.9): for correct sps $WN^{\neg e}$ implies SN (Theorem 3.2). One crucial step in the proof of Theorem 3.2 is Proposition 3.9: for correct sps non-erasing weak normalization coincides with non-erasing strong normalization. Observe that the slice β of Figure 12 gives a counter-example to this equivalence for sps which are not correct (which don't satisfy the AC condition of Definition 2.9): β can be normalized without applying erasing steps (in the precise sense of Definition 3.1), but it is not strongly normalizable. More in detail, this counter-example is due to (i) the presence of deadlocks (the cut t is not erased by one reduction step of β , but it becomes a deadlock), (ii) the failure of the confluence of \sim_{cut} for general sps (indeed one of the main ingredients in the proof of Proposition 3.9 is the confluence of (a labelled variant of) cut-elimination on correct sps, Theorem 3.12).

2.3 The AC property

Cut-elimination can be performed without any reference to correctness. However we noticed in the previous subsection that in presence of cyclic sps there are “bad” computations (even without additives and even for typable sps): the weak (and the strong) normalization property fails, so as the confluence property (recall the example of Figure 12).

We will then use in this paper the weakest (standard) notion of correctness known in the literature, requiring to our sps to be “switching acyclic” (or to satisfy AC). Switching acyclicity is required “slice by slice” (Definition 2.9). We use the AC condition to avoid cyclic sps (not enjoying WN), and to prove Theorem 3.12.

Definition 2.9 (AC condition) *A switching of a flat α is an (undirected) subgraph of α obtained by forgetting the orientation of α 's edges and by deleting one of the two premises of each \wp -node and $\text{?}c$ -node of α .*

We say that an sps π is switching acyclic (or satisfies AC) if every switching of every flat of π is an acyclic graph.

Examples of sps satisfying AC are the sps of Figures 8, 9, 10 and all sps pictured in Section 4.

The following proposition is an important property of sps, which will be used in this paper to prove SN and confluence of the *str*-reduction (Prop. 2.20 and Prop. 3.17):

Proposition 2.10 *Let π be an sps and x be a cut link of π which is not of type $(\top - cc)$ ¹⁴. If π satisfies AC and $\pi \rightsquigarrow_x \pi'$, then π' satisfies AC.*

PROOF. Standard (see [Dan90]). □

2.4 Two results of strong normalization

We now define two notable subreductions of \rightsquigarrow_{cut} : the *logical reduction* \rightsquigarrow_{log} and the *structural reduction* \rightsquigarrow_{str} (Definition 2.11). These two reductions are disjoint and their union is \rightsquigarrow_{cut} . A basic fact, crucial in the next Section 3, is that both \rightsquigarrow_{log} and \rightsquigarrow_{str} are SN on sps satisfying AC (Proposition 2.14 and Proposition 3.15), even if their union \rightsquigarrow_{cut} is not even WN (see the example of Fig. 10). More precisely, we

¹⁴One has to refuse $(\top - cc)$ steps, essentially because they are not “local”; for the same reason $(\top - cc)$ and $(ccad)$ steps of [Gir87a] and [Tdf03] can be performed only in presence of correctness, and of a much stronger notion of correctness than AC.

define two measures on sps, $|\pi|_{log}$ and $|\pi|_{str}$ (Definition 2.13 and Definition 2.18), and we prove that such measures shrink after every $\rightsquigarrow_{log}/\rightsquigarrow_{str}$ reduction step. In addition, we briefly discuss how the reduction \rightsquigarrow_{str} increases $|\pi|_{log}$ and conversely how the reduction \rightsquigarrow_{log} increases $|\pi|_{str}$, in accordance with the fact that $\rightsquigarrow_{cut} = \rightsquigarrow_{log} \cup \rightsquigarrow_{str}$ is not WN.

Let us stress that the *str*-measure is really sharp and it can be generalized to other notions of net (like for example differential nets [Pag08], [Tra08] or stream associative nets [PS08]).

Definition 2.11 *We define the following subreductions of \rightsquigarrow_{cut} :*

- *the logical reduction, denoted by \rightsquigarrow_{log} : a \rightsquigarrow_{log} -step is one of the \rightsquigarrow_{cut} -steps (ax) , (\otimes/\wp) , $(1/\perp)$, $(\oplus_i/\&_j)$, $(!/?d)$, and $(\top - cc)$;*
- *the structural reduction, denoted by \rightsquigarrow_{str} : a \rightsquigarrow_{str} -step is one of the \rightsquigarrow_{cut} -steps $(!/?w)$, $(!/?c)$ and $(!/!)$.*

Of course we have $\rightsquigarrow_{cut} = \rightsquigarrow_{log} \cup \rightsquigarrow_{str}$.

2.4.1 The strong normalization of \rightsquigarrow_{log}

We now prove that \rightsquigarrow_{log} enjoys strong normalization. Contrary to the case of \rightsquigarrow_{str} (where this property holds only for sps satisfying AC: recall for example the slice α of Fig. 11, which does not enjoy SN -and not even WN- and does not satisfy AC), we show that SN of \rightsquigarrow_{log} holds for *every* sps, regardless its correctness.

Consider an sps π which has at most one slice, and recursively s.t. every box of π has at most one slice. For such a π , the proof that \rightsquigarrow_{log} is SN is immediate: every step of \rightsquigarrow_{log} strictly decreases the number of links of π (and keeps the property of having at most one slice in each box). However, in the general case the links of an sps might increase after a \rightsquigarrow_{log} reduction: if π has a box π^o associated with a $!$ -link o and containing more than one slice, then a $(!/?d)$ step “opening” o will duplicate the links at the same depth as o , as many times as the number of slices in π^o . This means we need to find a sharper measure on sps than their sizes, in order to point out what is decreasing under \rightsquigarrow_{log} . In some sense, the previous remark concerning a very special case of sps can also be used in the general case, thanks to the notion of *single-threaded slice* introduced in [LTdF04]¹⁵.

Definition 2.12 (Single-threaded slice, [LTdF04]) A single-threaded slice, sgth

¹⁵ There are actually two differences w.r.t. [LTdF04]: in that paper the authors define the *set* of sgth of a proof-net, we are instead interested in the *multiset* of sgth of an sps (this difference is due to the fact that our sps are multisets of slices and not simply sets, as it is in [LTdF04]); the other difference is that while in [LTdF04] there is *exactly* one slice associated with every $!$ -link, we have here *at most* one slice (this difference is necessary to deal with the case of $!$ -links having an empty sps inside in Proposition 2.14).

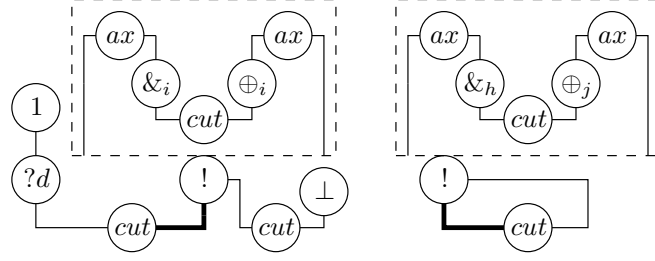


Fig. 13. The multiset $\text{sgth}(\pi)$, where π is the sps of Fig. 3, consists of four single-threaded slices, given by the above picture taking $i, h, j \in \{1, 2\}$, $h \neq j$.

for short, of depth 0 is simply a slice of depth 0; a single-threaded slice of depth $n + 1$ is a slice α of depth $n + 1$ s.t. with every $!$ -link at depth 0 of α is associated either the empty sps (with the appropriate conclusion) or a single-threaded slice (of depth at most n).

Given an sps π we define $\text{sgth}(\pi)$, the multiset of sgth of π , by induction on the depth of π :

- if π has depth 0, then $\text{sgth}(\pi) = \pi$ (which is a multiset of sgth);
- if π has depth $n + 1$, then $\alpha \in \text{sgth}(\pi)$ iff α is obtained from a slice α' of π by choosing for every $!$ -link o at depth 0 of α' :
 - the empty sps (with the appropriate conclusion), in case this sps is associated with o in π
 - a sgth of the sps associated with o , otherwise.

For an example see Fig. 13, which shows the sgth of the sps π of Fig. 3.

Definition 2.13 (Measure for \rightsquigarrow_{\log}) The log-measure of an sps π is a natural number, denoted by $|\pi|_{\log}$ and defined as follows:

$$|\pi|_{\log} := \sum_{\alpha \in \text{sgth}(\pi)} \text{size}(\alpha)$$

For example, consider the sps π of Fig. 3: we have $|\pi|_{\log} = 18 \times 4 = 72$. Notice that for every sps π , the log-measure of π is the sum of the log-measures of the slices of π , i.e. $|\pi|_{\log} = \sum_{\alpha \in \pi} |\alpha|_{\log}$.

Proposition 2.14 The reduction \rightsquigarrow_{\log} is SN on the sps.

PROOF. We prove that $\pi \rightsquigarrow_{\log} \bar{\pi}$ implies $|\pi|_{\log} > |\bar{\pi}|_{\log}$. Actually we can restrict to the case π has exactly one slice α . The general case can be deduced as follows. Suppose $\pi \rightsquigarrow_{\log} \bar{\pi}$, then there is a slice $\alpha \in \pi$ and a multiset $\pi_{\alpha} \subseteq \bar{\pi}$ s.t. $\alpha \rightsquigarrow_{\log} \pi_{\alpha}$

and $\pi \setminus [\alpha] = \bar{\pi} \setminus \pi_\alpha$ ¹⁶. By hypothesis we have that $|\alpha|_{log} > |\pi_\alpha|_{log}$, so:

$$\begin{aligned}
|\pi|_{log} &= |\pi \setminus [\alpha]|_{log} + |\alpha|_{log} \\
&= |\bar{\pi} \setminus \pi_\alpha|_{log} + |\alpha|_{log} \\
&> |\bar{\pi} \setminus \pi_\alpha|_{log} + |\pi_\alpha|_{log} \\
&= |\bar{\pi}|_{log}
\end{aligned}$$

So, we suppose $\alpha \rightsquigarrow_{log} \pi_\alpha$ and we prove $|\alpha|_{log} > |\pi_\alpha|_{log}$. The proof splits in the six cases associated with the types of the \rightsquigarrow_{log} steps: we treat in detail only the case of a (!/? d) cut at depth 0 in α , the other cases being immediate. Recall Fig. 4 and the notation of Def. 2.6 in the (!/? d) step. Let $[\beta_1, \dots, \beta_n]$ ($n \geq 0$) be the sps associated with the !-link l of α , so that $\pi_\alpha = [\alpha'_1, \dots, \alpha'_n]$. If $n = 0$, then π_α is empty (α is erased) and clearly $|\alpha|_{log} > |\pi_\alpha|_{log}$ ¹⁷. Otherwise, one can associate with every $i \leq n$ the multiset $\text{sgth}(\alpha)_i$ of sgth of α choosing the slice β_i for l (thus $\text{sgth}(\alpha)_i \subseteq \text{sgth}(\alpha)$). Notice that $\text{sgth}(\alpha) = \sum_{i \leq n} \text{sgth}(\alpha)_i$, so in particular $|\alpha|_{log} = \sum_{i \leq n} |\text{sgth}(\alpha)_i|_{log}$; moreover to every i and every $\gamma \in \text{sgth}(\alpha)_i$ it corresponds exactly one element $\gamma' \in \text{sgth}(\alpha'_i)$ and $\text{size}(\gamma) = \text{size}(\gamma') + 2$ (in γ' the !-link l and the ? d -link above t have been erased), so $|\text{sgth}(\alpha)_i|_{log} > |\alpha'_i|_{log}$. Since $|\pi_\alpha|_{log} = \sum_{i \leq n} |\alpha'_i|_{log}$, we conclude $|\alpha|_{log} > |\pi_\alpha|_{log}$. \square

Notice that the above proof uses the property that after a \rightsquigarrow_{log} step the number of elements of the multiset of sgth of an sps cannot increase: if $\pi \rightsquigarrow_{log} \bar{\pi}$, then $\text{sgth}(\bar{\pi})$ has at most the same cardinality as $\text{sgth}(\pi)$. It is remarkable that this property fails under \rightsquigarrow_{str} : for example, if $\pi \rightsquigarrow_{str} \bar{\pi}$ is obtained by a (!/? c) step then $\text{sgth}(\bar{\pi})$ might have more elements than $\text{sgth}(\pi)$, since we have duplicated the possible choices one has to do on the duplicated !-link in order to obtain $\text{sgth}(\bar{\pi})$. The increase of the number of sgth in $\bar{\pi}$ entails the increase of the log -measure: in the (!/? c) case, we can very well have $|\pi|_{log} < |\bar{\pi}|_{log}$.

2.4.2 The strong normalization of \rightsquigarrow_{str}

We now want to prove that \rightsquigarrow_{str} enjoys SN on sps satisfying AC (Prop. 2.20). As already mentioned in the Introduction, the delicate point behind Prop. 2.20 is that the reduction of a cut t may affect the reduction of other cuts, even at the same exponential depth as t . The critical pairs presented in the proof of Lemma 3.16

¹⁶ We use here the standard set notation for multisets of slices: for example $\pi \setminus [\alpha]$ is the multiset π of slices without one occurrence of the slice α .

¹⁷ This case motivates our variant Definition 2.12 of sgth : if exactly one slice were associated with every !-link, in this case we would have $\text{sgth}(\pi) = \text{sgth}(\bar{\pi}) = \emptyset$ and nothing would decrease.



Fig. 14. Behaviour of an exponential path meeting structural links (i.e. $!$ -, $?w$ -, $?c$ -links) and cuts (it stops on all the other links); on $?c$ -links a path can choose which premise to follow.

of the next Section 3 are examples of this “meddling” of a str -reduction step in another str -reduction step, the most evident example being the slice α of case 2 of the mentioned proof: in that case the reduction of the cut t even changes the type of the cut r from $(!/!)$ to $(!/?c)$.

This meddling makes delicate the proof of SN for \rightsquigarrow_{str} . Let us explain why this phenomenon is peculiar to linear logic and absent in λ -calculus: in λ -calculus the β -reduction of a redex may affect other redexes only if these last ones are deeper than the former one (in LL’s terminology: the affected redexes must have a greater exponential depth than that of the reduced one). Contrary to what happens in λ -calculus, in LL there are cuts at the same exponential depth which affect each other. Something similar to what we have here can be found only in λ -calculi with explicit substitution (see for example [DCKP03]).

The key ingredient we found to tame this meddling of cuts in the life of their fellows with the same exponential depth is the following notion of *exponential dependence* (Definition 2.15). Roughly speaking, exponential dependence has two main properties:

- given two cuts at the same exponential depth t and r , the str -reduction of t affects the str -reduction of r only if the premises of r “depend exponentially” on the premises of t ;
- exponential dependence is (in some sense) stable under str -reduction.

These properties are at the base of the proof of Proposition 2.20.

Definition 2.15 (Exponential dependence) *Let π be an sps, and let ϕ be a (n oriented) path in a flat of π . We say that ϕ is exponential when (see Figure 14):*

- ϕ crosses only structural links (i.e. links of type $!$, $?w$ or $?c$) and cuts;
- if ϕ crosses an edge c upward, then c is an edge (conclusion or premise) of a structural $?c$ -link (i.e. a link of type $?w$ or $?c$) or an auxiliary conclusion of a $!$ -link;
- if ϕ crosses an edge c downward, then c is the main conclusion of a $!$ -link and premise of a cut link.

Given two edges a and b , we say that a exponentially depends on b , whenever there

is an exponential path from a to b . Given an edge a , we denote by $Pred(a)$ the set of the immediate predecessors of a i.e. the set of those edges b such that there is an exponential path from a to b crossing exactly one node.

The following fact allows to make induction on the maximal length of the exponential paths starting from a given edge¹⁸ :

Fact 2.16 *If π satisfies AC, then every exponential path of π is finite.*

PROOF. Immediate consequence of the fact that every exponential path of π is a path of some switching of π (remember Definition 2.9): hence if π satisfies AC, then π has no exponential cycle. \square

Given an sps π satisfying AC, we define two functions (wd^π and ln^π) associating an integer with every edge a (resp. !-link o) of π . What happens (as the reader will see during the proof of Proposition 2.20), is that for every !-link o at depth 0 of π , on the one hand $wd^\pi(o)$ is an upper bound for the number of times o can be copied during a *str*-reduction sequence starting from π , and on the other hand $ln^\pi(o)$ is an upper bound for the depth of o 's residues during a *str*-reduction sequence starting from π . A first evidence that the length of the *str*-reduction sequences starting from an sps π is bounded, is the existence of these two functions: the function wd^π “measures” the maximal “width” of π 's reducts while the function ln^π “measures” the maximal depth of π 's reducts.

Definition 2.17 *Let π be an sps satisfying AC and let a be an edge of π (at any depth). We define $wd^\pi(a)$ (the width of a in π) and $ln^\pi(a)$ (the length of a in π) by double induction: the first component is $depth(\pi)$, the second one is the maximal length of the exponential paths of π starting from a .*

$$ln^\pi(a) := \begin{cases} 1 & \text{if } Pred(a) = \emptyset, \\ 1 + \max_{a' \in \rho(a)} (ln^\pi(a')) + ln^\pi(b) & \text{if } a \text{ is an auxiliary conclusion of a !-} \\ & \text{link with main conclusion } b, \text{ box } \rho \text{ and} \\ & \rho(a) \text{ is the set of the conclusions of the} \\ & \text{slices of } \rho \text{ corresponding to } a, \\ 1 + \max_{b \in Pred(a)} (ln^\pi(b)) & \text{otherwise.} \end{cases}$$

¹⁸ One could also make induction on the following well-founded partial order on the edges of an sps satisfying AC: $a \geq_{exp} b$ whenever there is an exponential path from a to b . However, the proof that \geq_{exp} is indeed a partial order is not immediate, and this paper does not lack delicate proofs...

$$\text{wd}^\pi(a) := \begin{cases} 1 & \text{if } \text{Pred}(a) = \emptyset, \\ \left(\sum_{a' \in \rho(a)} \text{wd}^\rho(a') \right) \text{wd}^\pi(b) & \text{if } a \text{ is an auxiliary conclusion of a !-} \\ & \text{link with main conclusion } b, \text{ box } \rho \text{ and} \\ & \rho(a) \text{ is the set of the conclusions of the} \\ & \text{slices of } \rho \text{ corresponding to } a, \\ \sum_{b \in \text{Pred}(a)} \text{wd}^\pi(b) & \text{otherwise.} \end{cases}$$

We extend wd^π and ln^π to the !-links of π , by setting for a !-link o with main conclusion b : $\text{ln}^\pi(o) = \text{ln}^\pi(b)$ and $\text{wd}^\pi(o) = \text{wd}^\pi(b)$.

We will often simply write $\text{ln}(a)$ or $\text{wd}(a)$ (resp. $\text{ln}(o)$ or $\text{wd}(o)$), when there is no ambiguity on the sps π we refer to.

Finally the long-awaited definition of *str*-measure:

Definition 2.18 (Measure for \rightsquigarrow_{str}) *The str-measure of an sps π satisfying AC is a natural number, denoted by $|\pi|_{str}$ and defined as follows (by induction on $\text{depth}(\pi)$):*

$$|\pi|_{str} := \sum_{o \in !(\pi)} (\text{wd}^\pi(o) \cdot (\text{ln}^\pi(o) + |\pi^o|_{str}))$$

where $!(\pi)$ denotes the set of !-links at depth 0 of π (and π^o denotes as usual the sps associated with the !-link o).

Consider for example the slice δ of Fig. 8, we have $\text{ln}^\delta(b) = 1$, $\text{ln}^\delta(a) = 1 + 1 + 1 = 3$ and $\text{ln}^\delta(d) = 1 + \max\{1, 3\} = 4$; as for the width, $\text{wd}^\delta(b) = \text{wd}^\delta(a) = 1$, $\text{wd}^\delta(d) = 2$. Thus $|\delta|_{str} = \text{wd}^\delta(b) \cdot (\text{ln}^\delta(b) + 0) = 1$. Then consider the slice $\delta\delta$ of Fig. 9, we have $\text{ln}^{\delta\delta}(e) = 1 + \text{ln}^{\delta\delta}(d) = 1 + \text{ln}^\delta(d) = 5$ and $\text{wd}^{\delta\delta}(e) = \text{wd}^{\delta\delta}(d) = \text{wd}^\delta(d) = 2$. So that $|\delta\delta|_{str} = 1 + \text{wd}^{\delta\delta}(e) (\text{ln}^{\delta\delta}(e) + |\delta|_{str}) = 13$. The reader can check that the *str*-reducts of $\delta\delta$ (i.e. the slices pictured in the above line of Fig. 10) have a *str*-measure strictly less than $|\delta\delta|_{str}$. Moreover, notice that the reduction $\delta\delta \rightsquigarrow_{cut}^+ \delta\delta$ contains *log*-steps (the 3 ones represented in the lower part of Fig. 10).

The following fact shows some kind of “modularity” and of “monotonicity” of the functions ln and wd , two ingredients of Proposition 2.20. We use the (rather intuitive) notion of module: it is a subgraph of a flat (definition 2.1) and can thus have edges without target as well as edges without source (the set of these edges is often called “the interface” of the module).

Recall that a slice is the data of a flat and a function associating with every !-link an sps: it is then always possible to treat a slice as a flat (just forget the function).

Fact 2.19 (Modularity) *Let α and α' be two switching acyclic slices and suppose that the flat α' is the flat α where the module β' substitutes the module β , as pictured below:*



If b is an edge of the interface of β , denote by b' the corresponding edge of the interface of β' . The following properties hold:

- (1) *let a be an edge of the interface of β and suppose that a exponentially depends only on edges of γ (i.e. if a depends on $c \neq a$, then c is not an edge of β). Then $\ln^\alpha(a) = \ln^{\alpha'}(a')$ and $\text{wd}^\alpha(a) = \text{wd}^{\alpha'}(a')$;*
- (2) *property (1) holds even in case some edges of the interface of β (resp. β') are among the conclusions of α (resp. α'). In this case, the interfaces of β and β' need not coincide: they do only on the edges that are not conclusions of α and α' ;*
- (3) *if d is an edge of α which is not an edge of β and if for every edge b of the interface of β we have $\ln^\alpha(b) = \ln^{\alpha'}(b')$ (resp. $\ln^\alpha(b) \geq \ln^{\alpha'}(b')$), then $\ln^\alpha(d) = \ln^{\alpha'}(d)$ (resp. $\ln^\alpha(d) \geq \ln^{\alpha'}(d)$); the same holds for wd .*

PROOF. It is a consequence of Definition 2.17. □

As for the *log*-measures, the *str*-measure of π is the sum of the *str*-measures of the slices of π , i.e. $|\pi|_{str} = \sum_{\alpha \in \pi} |\alpha|_{str}$.

Proposition 2.20 *The reduction \rightsquigarrow_{str} is SN on the sps satisfying AC.*

PROOF. For π satisfying AC, we prove that $\pi \rightsquigarrow_{str} \bar{\pi}$ implies $|\pi|_{str} > |\bar{\pi}|_{str}$. As for the proof of Prop. 2.14, we can restrict to the case π has exactly one slice α (the extension to the general case is obtained exactly as in the proof of Prop. 2.14).

So let α be a switching acyclic slice and $\alpha \rightsquigarrow_{str} \pi_\alpha$. First remark that π_α contains a unique slice (a glance at Figure 4 will convince the reader). Let then $\pi_\alpha = \{\alpha'\}$ and let t be the cut link of α reduced during the \rightsquigarrow_{str} step under consideration. We prove that:

- (1) $|\alpha|_{str} > |\alpha'|_{str}$,
- (2) for every conclusion d of α , let \vec{d} be the conclusion of α' corresponding to d , one has $\ln^\alpha(d) \geq \ln^{\alpha'}(\vec{d})$ and $\text{wd}^\alpha(d) \geq \text{wd}^{\alpha'}(\vec{d})$.

The proof is by induction of the depth of t .

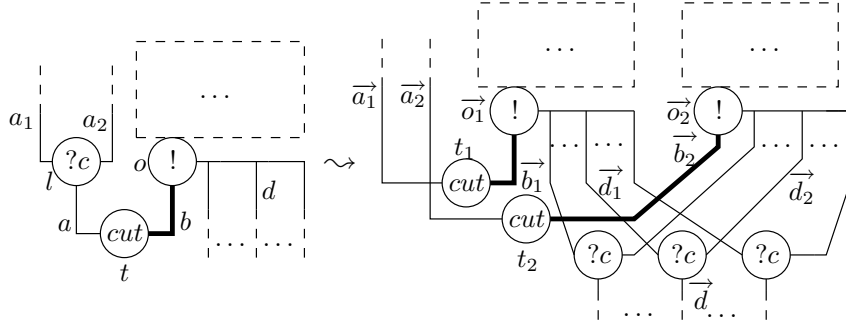


Fig. 15. The (!/?c) case of the proof of Proposition 2.20

Base of induction. If t has depth 0, then we have three cases, depending on the type of t .

Case (!/?c). If t is a cut of type (!/?c), then (see Figure 15) let l (resp. o) be the ?c-link (resp. !-link) whose conclusion a (resp. whose main conclusion b) is a premise of t , let a_1, a_2 be the premises of l , and let \vec{o}_1, \vec{o}_2 be the two copies of o in α' : \vec{o}_i and \vec{a}_i ($i \in \{1, 2\}$) are premises of a cut link t_i .

We set $!(\alpha) = \{o\} \cup S$, i.e. S is the set of !-links at depth 0 of α different from o . Observe that every !-link $v \in S$ has a unique residue \vec{v} in α' : we have $!(\alpha') = \{\vec{o}_1, \vec{o}_2\} \cup \vec{S}$. Let us define:

$$|S|_{str} := \sum_{v \in S} (\text{wd}^\alpha(v) \cdot (\ln^\alpha(v) + |\pi^v|_{str}))$$

$$|\vec{S}|_{str} := \sum_{\vec{v} \in \vec{S}} (\text{wd}^{\alpha'}(\vec{v}) \cdot (\ln^{\alpha'}(\vec{v}) + |\pi^{\vec{v}}|_{str})).$$

Then by definition we have:

$$|\alpha|_{str} = \text{wd}^\alpha(o)(\ln^\alpha(o) + |\pi^o|_{str}) + |S|_{str}$$

$$|\alpha'|_{str} = \text{wd}^{\alpha'}(\vec{o}_1) (\ln^{\alpha'}(\vec{o}_1) + \pi^{\vec{o}_1}) + \text{wd}^{\alpha'}(\vec{o}_2) (\ln^{\alpha'}(\vec{o}_2) + \pi^{\vec{o}_2}) + |\vec{S}|_{str}.$$

Consider any conclusion d of o in α different from b , denote as \vec{d} the conclusion of the ?c-link created in α' and corresponding to d , and denote as \vec{d}_1, \vec{d}_2 the two premises of this ?c-link, one being an auxiliary conclusion of \vec{o}_1 , the other one being an auxiliary conclusion of \vec{o}_2 (see Figure 15). We prove $\ln^\alpha(d) = \ln^{\alpha'}(\vec{d})$ and $\text{wd}^\alpha(d) = \text{wd}^{\alpha'}(\vec{d})$.

Note that the edge a_i ($i \in \{1, 2\}$) does not exponentially depend on any edge involved in the reduction of t , so by Fact 2.19-1, one has $\ln(\vec{a}_i) = \ln(a_i)$ and $\text{wd}(\vec{a}_i) = \text{wd}(a_i)$. Then the following equalities hold (where for $\gamma \in \pi^o$ (resp. $\gamma \in \pi^{\vec{o}_i}$) we denote by d^γ the conclusion of γ corresponding to d):

$$\begin{aligned}
\ln^\alpha(d) &= 1 + \max_{\gamma \in \pi^o} (\ln^\gamma(d^\gamma)) + \ln^\alpha(b) \\
&= 2 + \max_{\gamma \in \pi^o} (\ln^\gamma(d^\gamma)) + \ln^\alpha(a) \\
&= 3 + \max_{\gamma \in \pi^o} (\ln^\gamma(d^\gamma)) + \max\{\ln(a_1), \ln(a_2)\} \\
&= 1 + \max \left\{ \begin{array}{l} \left(\max_{\gamma \in \pi^{\vec{o}_1}} (\ln^\gamma(d^\gamma)) + \ln(\vec{a}_1) + 2 \right), \\ \left(\max_{\gamma \in \pi^{\vec{o}_2}} (\ln^\gamma(d^\gamma)) + \ln(\vec{a}_2) + 2 \right) \end{array} \right\} \\
&= 1 + \max\{\ln^{\alpha'}(\vec{d}_1), \ln^{\alpha'}(\vec{d}_2)\} \\
&= \ln^{\alpha'}(\vec{d}).
\end{aligned}$$

As for $\text{wd}^\alpha(d) = \text{wd}^{\alpha'}(\vec{d})$:

$$\begin{aligned}
\text{wd}^\alpha(d) &= \left(\sum_{\gamma \in \pi^o} \text{wd}^\gamma(d^\gamma) \right) \cdot \text{wd}^\alpha(b) \\
&= \left(\sum_{\gamma \in \pi^o} \text{wd}^\gamma(d^\gamma) \right) \cdot \text{wd}^\alpha(a) \\
&= \left(\sum_{\gamma \in \pi^o} \text{wd}^\gamma(d^\gamma) \right) \cdot (\text{wd}(a_1) + \text{wd}(a_2)) \\
&= \left(\sum_{\gamma \in \pi^{\vec{o}_1}} \text{wd}^\gamma(d^\gamma) \right) \cdot \text{wd}(\vec{a}_1) + \left(\sum_{\gamma \in \pi^{\vec{o}_2}} \text{wd}^\gamma(d^\gamma) \right) \cdot \text{wd}(\vec{a}_2) \\
&= \left(\sum_{\gamma \in \pi^{\vec{o}_1}} \text{wd}^\gamma(d^\gamma) \right) \cdot \text{wd}(\vec{b}_1) + \left(\sum_{\gamma \in \pi^{\vec{o}_2}} \text{wd}^\gamma(d^\gamma) \right) \cdot \text{wd}(\vec{b}_2) \\
&= \text{wd}(\vec{d}_1) + \text{wd}(\vec{d}_2) \\
&= \text{wd}^{\alpha'}(\vec{d}).
\end{aligned}$$

This implies by Fact 2.19-3, that for every $v \in S$ one has $\ln(v) = \ln(\vec{v})$ and $\text{wd}(v) = \text{wd}(\vec{v})$, and that for every conclusion d of α , $\ln^\alpha(d) = \ln^{\alpha'}(\vec{d})$ and $\text{wd}^\alpha(d) = \text{wd}^{\alpha'}(\vec{d})$, in particular we have condition 2.

As for condition 1, notice that for every $v \in S$ the box associated with v in π is the same sps as the box associated with \vec{v} in α' , so that $|\pi^v|_{str} = |\pi^{\vec{v}}|_{str}$. This means that $|S|_{str} = |\vec{S}|_{str}$. We can thus reduce the inequality $|\alpha|_{str} > |\alpha'|_{str}$ to the following one:

$$\begin{aligned}
\text{wd}(o)(\ln(o) + |\pi^o|_{str}) &> \text{wd}(\vec{o}_1)(\ln(\vec{o}_1) + |\pi^{\vec{o}_1}|_{str}) + \\
&\quad + \text{wd}(\vec{o}_2)(\ln(\vec{o}_2) + |\pi^{\vec{o}_2}|_{str}).
\end{aligned}$$

Notice that $\ln(o) = 1 + \max\{\ln(\vec{o}_1), \ln(\vec{o}_2)\}$, so that $\ln(o) > \ln(\vec{o}_i)$ ($i \in \{1, 2\}$). Notice also that $\text{wd}(o) = \text{wd}(\vec{o}_1) + \text{wd}(\vec{o}_2)$, as well as $\pi^o = \pi^{\vec{o}_i}$ ($i \in \{1, 2\}$). From these remarks the above inequality easily follows. We conclude $|\alpha|_{str} > |\alpha'|_{str}$.

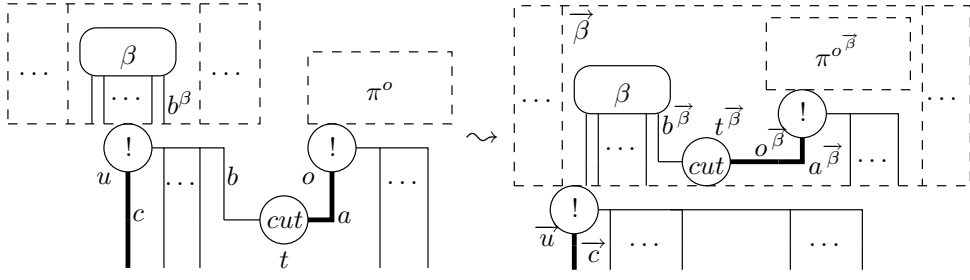


Fig. 16. The (!/!) case of the proof of Proposition 2.20

Case (!/!). If t is of type (!/!), then (see Figure 16) let o (resp. u) be the !-link of which the main (resp. an auxiliary) conclusion a (resp. b) is a premise of t , let c be the main conclusion of u , let \vec{u} be the residue of u in α' and for every slice $\vec{\beta} \in \pi^{\vec{u}}$, let $o^{\vec{\beta}}$ be the copy of o which “entered” the !-link u during the reduction of t and finally let $t^{\vec{\beta}}$ be the cut created in $\vec{\beta}$ by the reduction of t .

We set $!(\alpha) = \{u, o\} \cup S$, i.e. S is the set of !-links at depth 0 of α different from u and o . Observe that each !-link $v \in S$ has a unique residue \vec{v} in α' , so that $!(\alpha') = \{\vec{u}\} \cup \vec{S}$. Let us define:

$$|S|_{str} := \sum_{v \in S} (\text{wd}^\alpha(v) \cdot (\ln^\alpha(v) + |\pi^v|_{str}))$$

$$|\vec{S}|_{str} := \sum_{\vec{v} \in \vec{S}} (\text{wd}^{\alpha'}(\vec{v}) \cdot (\ln^{\alpha'}(\vec{v}) + |\pi^{\vec{v}}|_{str})).$$

Then by definition we have:

$$|\alpha|_{str} = \text{wd}(u) (\ln(u) + |\pi^u|_{str}) + \text{wd}(o) (\ln(o) + |\pi^o|_{str}) + |S|_{str}$$

$$|\alpha'|_{str} = \text{wd}(\vec{u}) (\ln(\vec{u}) + |\pi^{\vec{u}}|_{str}) + |\vec{S}|_{str}.$$

Consider any conclusion d of u in α different from b , and denote as \vec{d} the conclusion of \vec{u} in α' corresponding to d . If $d = c$, then d does not exponentially depend on any edge involved in the reduction of t : so by Fact 2.19-1, one has $\ln^\alpha(c) = \ln^{\alpha'}(\vec{c})$. Otherwise, let π^u (resp. $\pi^{\vec{u}}$) be the box associated with u (resp. \vec{u}). By Definition 2.17 (with the same notations), one has:

$$\ln^\alpha(d) := 1 + \max_{a \in \pi^u(d)} (\ln^{\pi^u}(a)) + \ln^\alpha(c)$$

$$\ln^{\alpha'}(\vec{d}) := 1 + \max_{a \in \pi^{\vec{u}}(\vec{d})} (\ln^{\pi^{\vec{u}}}(a)) + \ln^{\alpha'}(\vec{c}).$$

Notice that every edge g conclusion of π^u exponentially depends only on edges which are not involved in the reduction of t , so by Fact 2.19-2 we have $\ln^{\pi^u}(g) = \ln^{\pi^{\vec{u}}}(\vec{g})$. Since we already noticed that $\ln^\alpha(c) = \ln^{\alpha'}(\vec{c})$, we can eventually conclude that for every conclusion d of u in π different from b one has $\ln(d) = \ln(\vec{d})$,

and in the same way one obtains the equality $\text{wd}(d) = \text{wd}(\vec{d})$. (In particular, $\ln(u) = \ln(\vec{u})$ and $\text{wd}(u) = \text{wd}(\vec{u})$.)

Then, consider any auxiliary conclusion d of o in α , and denote by \vec{d} the conclusion of \vec{u} in α' corresponding to d . Also in this case we will prove $\ln^\alpha(d) = \ln^{\alpha'}(\vec{d})$ and $\text{wd}^\alpha(d) = \text{wd}^{\alpha'}(\vec{d})$. Still by Fact 2.19-2 we have that $\ln^\beta(b^\beta) = \ln^{\vec{\beta}}(b^{\vec{\beta}})$. The following equalities hold (where, like in the (!/?c)-case, d^γ denotes the conclusion of the slice γ corresponding to d):

$$\begin{aligned}
\ln^\alpha(d) &= 1 + \max_{\gamma \in \pi^o} (\ln^\gamma(d^\gamma)) + \ln^\alpha(a) \\
&= 2 + \max_{\gamma \in \pi^o} (\ln^\gamma(d^\gamma)) + \ln^\alpha(b) \\
&= 3 + \max_{\gamma \in \pi^o} (\ln^\gamma(d^\gamma)) + \max_{\beta \in \pi^u} (\ln^\beta(b^\beta)) + \ln^\alpha(c) \\
&= 3 + \max_{\gamma \in \pi^o} (\ln^\gamma(d^\gamma)) + \max_{\vec{\beta} \in \pi^{\vec{u}}} (\ln^{\vec{\beta}}(b^{\vec{\beta}})) + \ln^{\alpha'}(\vec{c}) \\
&= 3 + \max_{\vec{\beta} \in \pi^{\vec{u}}} \left(\max_{\gamma \in \pi^o \vec{\beta}} (\ln^\gamma(d^\gamma)) + \ln^{\vec{\beta}}(b^{\vec{\beta}}) \right) + \ln^{\alpha'}(\vec{c}) \\
&= 2 + \max_{\vec{\beta} \in \pi^{\vec{u}}} \left(\max_{\gamma \in \pi^o \vec{\beta}} (\ln^\gamma(d^\gamma)) + \ln^{\vec{\beta}}(a^{\vec{\beta}}) \right) + \ln^{\alpha'}(\vec{c}) \\
&= 1 + \max_{\vec{\beta} \in \pi^{\vec{u}}} (\ln^{\vec{\beta}}(d^{\vec{\beta}})) + \ln^{\alpha'}(\vec{c}) \\
&= \ln^{\alpha'}(\vec{d}).
\end{aligned}$$

In a similar way we prove that $\text{wd}^\alpha(d) = \text{wd}^{\alpha'}(\vec{d})$:

$$\begin{aligned}
\text{wd}^\alpha(d) &= \left(\sum_{\gamma \in \pi^o} \text{wd}^\gamma(d^\gamma) \right) \cdot \text{wd}^\alpha(a) \\
&= \left(\sum_{\gamma \in \pi^o} \text{wd}^\gamma(d^\gamma) \right) \cdot \text{wd}^\alpha(b) \\
&= \left(\sum_{\gamma \in \pi^o} \text{wd}^\gamma(d^\gamma) \right) \cdot \left(\sum_{\beta \in \pi^u} \text{wd}^\beta(b^\beta) \right) \cdot \text{wd}^\alpha(c) \\
&= \left(\sum_{\gamma \in \pi^o} \text{wd}^\gamma(d^\gamma) \right) \cdot \left(\sum_{\vec{\beta} \in \pi^{\vec{u}}} \text{wd}^{\vec{\beta}}(b^{\vec{\beta}}) \right) \cdot \text{wd}^{\alpha'}(\vec{c}) \\
&= \left(\sum_{\vec{\beta} \in \pi^{\vec{u}}} \left(\sum_{\gamma \in \pi^o \vec{\beta}} \text{wd}^\gamma(d^\gamma) \right) \cdot \text{wd}^{\vec{\beta}}(b^{\vec{\beta}}) \right) \cdot \text{wd}^{\alpha'}(\vec{c}) \\
&= \left(\sum_{\vec{\beta} \in \pi^{\vec{u}}} \left(\sum_{\gamma \in \pi^o \vec{\beta}} \text{wd}^\gamma(d^\gamma) \right) \cdot \text{wd}^{\vec{\beta}}(a^{\vec{\beta}}) \right) \cdot \text{wd}^{\alpha'}(\vec{c}) \\
&= \left(\sum_{\vec{\beta} \in \pi^{\vec{u}}} \text{wd}^{\vec{\beta}}(d^{\vec{\beta}}) \right) \cdot \text{wd}^{\alpha'}(\vec{c}) \\
&= \text{wd}^{\alpha'}(\vec{d}).
\end{aligned}$$

Like in the (!/?c)-case, this implies by Fact 2.19-3, that for every $v \in S$ one has $\ln(v) = \ln(\vec{v})$ and $\text{wd}(v) = \text{wd}(\vec{v})$, and that for every conclusion d of α , $\ln^\alpha(d) = \ln^{\alpha'}(\vec{d})$ and $\text{wd}^\alpha(d) = \text{wd}^{\alpha'}(\vec{d})$, in particular we have condition 2.

As for condition 1, notice that for every $v \in S$ the box associated with v in α is the same sps as the box associated with \vec{v} in α' , so that $|S|_{str} = |\vec{S}|_{str}$. Thus the inequality $|\alpha|_{str} > |\alpha'|_{str}$ can be reduced to the following one:

$$\text{wd}(u) (\ln(u) + |\pi^u|_{str}) + \text{wd}(o) (\ln(o) + |\pi^o|_{str}) > \text{wd}(\vec{u}) (\ln(\vec{u}) + |\pi^{\vec{u}}|_{str}).$$

We know that $\text{wd}(u) = \text{wd}(\vec{u})$ and $\ln(u) = \ln(\vec{u})$. Moreover we have:

$$\begin{aligned} |\pi^{\vec{u}}|_{str} &= \sum_{\vec{\beta} \in \pi^{\vec{u}}} \left(|\beta|_{str} + \text{wd}^{\vec{\beta}}(o^{\vec{\beta}}) \left(\ln^{\vec{\beta}}(o^{\vec{\beta}}) + |\pi^{o^{\vec{\beta}}}|_{str} \right) \right) \\ &= |\pi^u|_{str} + \sum_{\vec{\beta} \in \pi^{\vec{u}}} \left(\text{wd}^{\vec{\beta}}(o^{\vec{\beta}}) \left(\ln^{\vec{\beta}}(o^{\vec{\beta}}) + |\pi^{o^{\vec{\beta}}}|_{str} \right) \right). \end{aligned}$$

This allows to reduce the inequality $|\alpha|_{str} > |\alpha'|_{str}$ to:

$$\text{wd}(o) (\ln(o) + |\pi^o|_{str}) > \text{wd}(\vec{u}) \sum_{\vec{\beta} \in \pi^{\vec{u}}} \left(\text{wd}^{\vec{\beta}}(o^{\vec{\beta}}) \left(\ln^{\vec{\beta}}(o^{\vec{\beta}}) + |\pi^{o^{\vec{\beta}}}|_{str} \right) \right).$$

By definition $\ln(o) = 2 + \ln(u) + \max_{\beta \in \pi^u} (\ln^\beta(b^\beta))$, so we have (using Fact 2.19):

$$\begin{aligned} \text{wd}(o) (\ln(o) + |\pi^o|_{str}) &\geq \text{wd}(o) \left(2 + \max_{\beta \in \pi^u} (\ln^\beta(b^\beta)) + |\pi^o|_{str} \right) \\ &= \left(\text{wd}(u) \sum_{\beta \in \pi^u} \text{wd}^\beta(b^\beta) \right) \left(2 + \max_{\beta \in \pi^u} (\ln^\beta(b^\beta)) + |\pi^o|_{str} \right) \\ &= \text{wd}(u) \sum_{\beta \in \pi^u} \left(\text{wd}^\beta(b^\beta) \left(2 + \max_{\beta \in \pi^u} (\ln^\beta(b^\beta)) + |\pi^o|_{str} \right) \right) \\ &\geq \text{wd}(u) \sum_{\beta \in \pi^u} \left(\text{wd}^\beta(b^\beta) \left(2 + \ln^\beta(b^\beta) + |\pi^o|_{str} \right) \right) \\ &= \text{wd}(\vec{u}) \sum_{\vec{\beta} \in \pi^{\vec{u}}} \left(\text{wd}^{\vec{\beta}}(b^{\vec{\beta}}) \left(2 + \ln^{\vec{\beta}}(b^{\vec{\beta}}) + |\pi^{o^{\vec{\beta}}}|_{str} \right) \right) \\ &= 1 + \text{wd}(\vec{u}) \sum_{\vec{\beta} \in \pi^{\vec{u}}} \left(\text{wd}^{\vec{\beta}}(o^{\vec{\beta}}) \left(\ln^{\vec{\beta}}(o^{\vec{\beta}}) + |\pi^{o^{\vec{\beta}}}|_{str} \right) \right) \\ &> \text{wd}(\vec{u}) \sum_{\vec{\beta} \in \pi^{\vec{u}}} \left(\text{wd}^{\vec{\beta}}(o^{\vec{\beta}}) \left(\ln^{\vec{\beta}}(o^{\vec{\beta}}) + |\pi^{o^{\vec{\beta}}}|_{str} \right) \right). \end{aligned}$$

We thus conclude $|\alpha|_{str} > |\alpha'|_{str}$.

Case (!/?w). If t is of type (!/?w), then the case is simple and left to the reader. Notice that in this case the values of length and width can decrease, since the reduction of t erases a !-link.

Inductive step. If t is a cut in a !-link o of α' , then α' is obtained by replacing the box π^o associated with o with a box $\overline{\pi^o}$ s.t. $\pi^o \rightsquigarrow_{str} \overline{\pi^o}$. By induction hypothesis we know that (1) $|\overline{\pi^o}|_{str} < |\pi^o|_{str}$, and (2) for every conclusion d of π^o , $\ln^{\overline{\pi^o}}(d) \geq \ln^{\pi^o}(d)$, $\text{wd}^{\overline{\pi^o}}(d) \geq \text{wd}^{\pi^o}(d)$.

By Fact 2.19-3, this implies the conditions 1 and 2 for α, α' . □

We conclude the subsection by noting that (in a perfectly symmetric way w.r.t. the *log*-measure) the *str*-measure may increase under \rightsquigarrow_{log} : indeed, the reduction \rightsquigarrow_{log} can change the exponential paths of an *sps* and their lengths may increase (think for example of the main conclusion of a *!*-link o which is the premise of a *?d*-link itself cut with a *!*-link u : after the reduction of the $(!/?d)$ cut, some exponential paths disappear and some new ones starting from o and crossing edges of u 's box might appear and might be longer than the erased ones: this is exactly what happens in the last reduction step of Fig. 10). If the length of exponential paths grows, so do also the values of the functions *ln* and *wd*, and consequently the *str*-measure.

3 Standardization for sliced pure structures

In this section we prove our main result: the standardization theorem for sps (Theorem 3.2). Basically, the standardization theorem reduces the problem of SN to a problem of WN w.r.t. a subreduction of \rightsquigarrow_{cut} , called *non-erasing reduction*. The non-erasing reduction steps have a key property: they never erase cuts different from the reduced one¹⁹. We are thus going to distinguish, among reducible cuts, those cuts whose reduction can erase other cuts (called erasing) from the others (called non erasing):

Definition 3.1 (Erasing cut) *A cut link l is erasing when (exactly) one of the following holds:*

- l is of type $(\oplus_i/\&_j)$ for $i \neq j$, $(!/?w)$ or $(\top - cc)$,
- l is of type $(!/?d)$ (resp. $(!/?p)$) and the empty sps is associated with the $!$ -link whose main conclusion (resp. auxiliary conclusion) is a premise of l .

The erasing reduction (resp. non-erasing reduction) is the restriction of \rightsquigarrow_{cut} to the erasing (resp. non-erasing) steps, and it is denoted by \rightsquigarrow_e (resp. \rightsquigarrow_{-e}).

Theorem 3.2 (Standardization for sps) *Let π be an sps which satisfies AC. If $\pi \in \text{WN}^{-e}$, then $\pi \in \text{SN}$.*

With respect to the description of Gandy’s method given in the introduction, the standardization theorem achieves all the tasks except the proof of WN^{-e} . We split the proof of Theorem 3.2 in two parts: in Subsection 3.1 we prove that SN is a consequence of SN^{-e} (Proposition 3.5), in Subsection 3.2 we prove the equivalence between SN^{-e} and WN^{-e} (Proposition 3.9). This last step is the most delicate one, and uses the key notion of labelled sps and labelled reduction (Definition 3.6 and Definition 3.7).

3.1 SN is a consequence of SN^{-e}

In this subsection we prove Proposition 3.5: SN is a consequence of SN^{-e} . The proof is based on two simple facts: (i) erasing steps can always be postponed after non-erasing ones (Lemma 3.4); (ii) there is no infinite sequence of erasing steps (the \rightsquigarrow_e reduction clearly decreases the size of an sps). Then suppose there exists

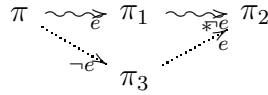
¹⁹The notion of non-erasing reduction has to be defined *very* carefully: not only a non-erasing reduction step does not erase cuts, but also it does not erase (nor change the non-erasing nature of) “future cuts”, that is of cuts which can be created during any reduction sequence. This might sound a bit mysterious by now, but it will appear in full light in Subsection 4.2.

an infinite sequence of \rightsquigarrow_{cut} steps starting from an sps π (i.e. suppose $\pi \notin SN$). This sequence should contain an infinite number of non-erasing steps: by iterating Lemma 3.4 we then obtain an arbitrarily long sequence of non-erasing steps starting from π , i.e. $\pi \notin SN^{-e}$.

Lemma 3.3 *Let π and π' be two sps. If $\pi \rightsquigarrow_e \pi'$, then every cut link t' of π' has an ancestor $\overleftarrow{t'}$ in π . If t' is non erasing, the two cuts t' and $\overleftarrow{t'}$ have the same type.*

PROOF. The only links which might be created by an erasing reduction step are $?w$ -links (see Definition 2.8): t' and $\overleftarrow{t'}$ might have a different type only if t' is erasing. \square

Lemma 3.4 (Postponement of \rightsquigarrow_e) *The reduction \rightsquigarrow_e can be postponed w.r.t. the reduction \rightsquigarrow_{-e} , i.e.:*



PROOF. Let t (resp. u) be the cut reduced in $\pi_1 \rightsquigarrow_{-e} \pi_2$ (resp. in $\pi \rightsquigarrow_e \pi_1$). By Lemma 3.3 and by the hypothesis that $\pi \rightsquigarrow_e \pi_1$, we deduce that t has an ancestor \overleftarrow{t} in π and that t and \overleftarrow{t} have the same type: in particular \overleftarrow{t} is non-erasing. We define π_3 as the result of reducing \overleftarrow{t} in π (so $\pi \rightsquigarrow_{-e} \pi_3$). By inspection of cases one can check that reducing the residues of u (which are all still erasing) in π_3 yields π_2 , i.e. $\pi_3 \rightsquigarrow_e^* \pi_2$. \square

Proposition 3.5 *Let π be an sps. If $\pi \in SN^{-e}$ then $\pi \in SN$.*

PROOF. Suppose that $\pi \notin SN$ and consider an infinite reduction sequence starting from π : $R := \pi \rightsquigarrow_{cut} \pi_1 \rightsquigarrow_{cut} \pi_2 \rightsquigarrow_{cut} \dots$. First remark that in R there is not an infinite suffix of erasing reduction steps. Indeed, the number of links strictly decreases when performing an erasing step.

We will define for any number n a sequence Q of \rightsquigarrow_{-e} steps of length n starting from π , hence proving that $\pi \notin SN^{-e}$. Let k be the first number (if any) s.t. $\pi_k \rightsquigarrow_e \pi_{k+1}$. We define Q by induction on $n - k$.

If $k \geq n$ or k does not exist, then simply take Q as the prefix of R of length n . If $k < n$, let m be the first integer s.t. $m > k$ and $\pi_m \rightsquigarrow_{-e} \pi_{m+1}$ (this m does exist since in R there is not an infinite suffix of erasing reduction steps). Apply $m - k$ times Lemma 3.4, in order to obtaining an (infinite) sequence of reductions R' which has a prefix of length $k + 1$ of \rightsquigarrow_{-e} steps. We obtain Q by applying the induction hypothesis to R' . \square

3.2 SN^{-e} is a consequence of WN^{-e}

This subsection is devoted to prove Proposition 3.9: the weak and strong normalization of \rightsquigarrow_{-e} are the same property for sps satisfying AC. This result is quite delicate and it is based on a confluence theorem (Theorem 3.12). Here is the more intriguing step of Gandy’s method: the basic idea is to find a measure $|\pi|_\ell$ on sps which is a natural number and to prove that (i) $|\pi|_\ell$ strictly increases under \rightsquigarrow_{-e} , (ii) \rightsquigarrow_{-e} is confluent. Then if an sps π has a normal form π' (i.e. if $\pi \in \text{WN}^{-e}$), we can deduce that the number $|\pi'|_\ell$ maximizes the length of every \rightsquigarrow_{-e} reduction sequence starting from π (and thus $\pi \in \text{SN}^{-e}$).

In order to define this increasing measure, we change a bit the syntax of sps, defining the labelled sps (Definition 3.6) and the labelled cut-elimination (Definition 3.7). The labelling plays the role of “counting” the number of steps applied to an sps, thus allowing the definition of an increasing measure (Definition 3.8).

Our guidelines in the definition of labelled sps and of their cut-elimination are the following: (i) labels will be used to define the measure which has to increase under cut-elimination (ii) to apply Gandy’s method we need confluence for *labelled* sps. We then introduce “new” links (actually dummy links) whose unique role is to participate (by means of their labels) to the measure of the sps: we call them $*$ -links. These dummy links are the only labelled ones. The idea is to associate with every maximal flat of an sps a unique $*$ -link, whose label is actually “the label of the flat”.

Definition 3.6 (Labelled sps) A $*$ -link is a link without premises nor conclusions. A labelled sliced pure structure, s^ℓ ps for short, is a couple $\langle \pi, \ell \rangle$ s.t. π is an sps with exactly one $*$ -link in every maximal flat of π , and ℓ is a function from the $*$ -links of π to the natural numbers, which is the labelling of $\langle \pi, \ell \rangle$.

The degree of the labelling of $\langle \pi, \ell \rangle$, denoted by $|\ell|$, is the sum of ℓ ’s values on the $*$ -links of π .

Definition 3.7 (Cut elimination for s^ℓ ps) Let t be a cut of $\langle \pi, \ell \rangle$. We allow to reduce t only in case it is non erasing. The result of the reduction of t is the s^ℓ ps $\langle \pi', \ell' \rangle$ defined as follows:

1. π' is the result of the reduction of t in π defined as in Definition 2.6, except if t is of type (!/?d): in that case, first erase the $*$ -link of the flat of t and then proceed like in Definition 2.6; in such a way, every maximal flat of π' contains exactly one $*$ -link;
2. let r be a $*$ -link of π' , we define $\ell'(r)$. Let \overleftarrow{r} be the ancestor of r in π ²⁰. There are three possible cases:

²⁰ The definition of ancestor/residue of a $*$ -link is straightforward, from the definition of π' given in step 1: every $*$ -link of π' has exactly one ancestor in π .

- (a) in case t is of type (ax) , $(1/\perp)$, (\otimes/\otimes) or $(\oplus_i/\&_i)$, if the flat of \overleftarrow{r} is the same as the flat of t , set $\ell'(r) = \ell(\overleftarrow{r}) + 1$; otherwise set $\ell'(r) = \ell(\overleftarrow{r})$;
- (b) in case t is of type $(!/?d)$, let v (resp. o) be the $*$ -link (resp. $!$ -link) in π erased by the reduction of t . If \overleftarrow{r} has depth 0 in the sps associated with o , then set $\ell'(r) = \ell(\overleftarrow{r}) + \ell(v) + 2$; otherwise set $\ell'(r) = \ell(\overleftarrow{r})$;
- (c) in case t is of type $(!/?c)$ or $(!/?p)$, set $\ell'(r) = \ell(\overleftarrow{r})$.

We introduce the notation $t(\langle \pi, \ell \rangle)$ as usual.

Definition 3.8 (Measure for \rightsquigarrow_ℓ) Let $\langle \pi, \ell \rangle$ be an s^ℓ ps, c be the number of $!$ -links of π , d be the sum of the depths of the $!$ -links of $\langle \pi, \ell \rangle$. The ℓ -measure of $\langle \pi, \ell \rangle$ is a natural number, denoted by $|\langle \pi, \ell \rangle|_\ell$ and defined as follows:

$$|\langle \pi, \ell \rangle|_\ell := (|\ell| + c)^2 + d$$

As we wrote at the beginning of the subsection, the reduction \rightsquigarrow_ℓ has two key properties: (i) it strictly increases the ℓ -measure (Lemma 3.10) and (ii) it is confluent (Theorem 3.12). These properties are used in the proof of Proposition 3.9:

Proposition 3.9 Let π be an sps which satisfies AC. If $\pi \in \text{WN}^{-e}$ then $\pi \in \text{SN}^{-e}$.

PROOF. Suppose π is an sps satisfying AC and s.t. $\pi \in \text{WN}^{-e}$. For every ℓ , $\langle \pi, \ell \rangle$ satisfies AC and $\langle \pi, \ell \rangle \in \text{WN}^\ell$. We prove that $\langle \pi, \ell \rangle \in \text{SN}^\ell$, which clearly implies that $\pi \in \text{SN}^{-e}$.

Let $\langle \pi', \ell' \rangle$ be a normal form of $\langle \pi, \ell \rangle$: we prove that the length of every reduction sequence starting from $\langle \pi, \ell \rangle$ is bounded by $|\langle \pi', \ell' \rangle|_\ell$.

Suppose $\langle \pi, \ell \rangle \rightsquigarrow_\ell \langle \pi_1, \ell_1 \rangle \rightsquigarrow_\ell \dots \rightsquigarrow_\ell \langle \pi_n, \ell_n \rangle$. By Lemma 3.10 we know that $|\langle \pi_n, \ell_n \rangle|_\ell \geq |\langle \pi, \ell \rangle|_\ell + n$. Since $\langle \pi, \ell \rangle$ satisfies AC we deduce by confluence (Theorem 3.12) that $\langle \pi_n, \ell_n \rangle \rightsquigarrow_\ell^* \langle \pi', \ell' \rangle$, hence by Lemma 3.10, $|\langle \pi', \ell' \rangle|_\ell \geq |\langle \pi_n, \ell_n \rangle|_\ell$. We then conclude that $|\langle \pi', \ell' \rangle|_\ell \geq n$. \square

Lemma 3.10 Let $\langle \pi, \ell \rangle$ be an s^ℓ ps: if $\langle \pi, \ell \rangle \rightsquigarrow_\ell \langle \pi', \ell' \rangle$, then $|\langle \pi', \ell' \rangle|_\ell > |\langle \pi, \ell \rangle|_\ell$.

PROOF. In case $\langle \pi', \ell' \rangle$ is the result of an (ax) , $(1/\perp)$, (\otimes/\otimes) or $(\oplus_i/\&_i)$ step, then $|\ell'| = |\ell| + 1$, $c' = c$, $d' = d$, hence $|\langle \pi', \ell' \rangle|_\ell > |\langle \pi, \ell \rangle|_\ell$.

In case $\langle \pi', \ell' \rangle$ is the result of a $(!/?c)$ step, then $|\ell'| \geq |\ell|$, $c' \geq c + 1$, $d' \geq d$, hence $|\langle \pi', \ell' \rangle|_\ell > |\langle \pi, \ell \rangle|_\ell$.

In case $\langle \pi', \ell' \rangle$ is the result of a $(!/?p)$ step, then $|\ell'| \geq |\ell|$, $c' \geq c$, $d' \geq d + 1$, hence $|\langle \pi', \ell' \rangle|_\ell > |\langle \pi, \ell \rangle|_\ell$.

In case $\langle \pi', \ell' \rangle$ is the result of a $(!/?d)$ step, then $|\ell'| \geq |\ell| + 2$, $c' \geq c - 1$, $d' \geq d - c + 1$. We deduce: $|\langle \pi', \ell' \rangle|_\ell \geq (|\ell| + c + 1)^2 + d - c + 1 \geq |\langle \pi, \ell \rangle|_\ell + c + 1$. \square

3.2.1 Confluence of \rightsquigarrow_ℓ

The rest of this section is devoted to the proof of confluence of ℓ -reduction (Theorem 3.12). Our ℓ -reduction (as well as usual LL cut-reduction) is locally confluent but not strongly confluent: it is not the case that $\langle \pi, \ell \rangle \rightsquigarrow_\ell \langle \pi_1, \ell_1 \rangle$ and $\langle \pi, \ell \rangle \rightsquigarrow_\ell \langle \pi_2, \ell_2 \rangle$ imply the existence of an s^ℓ ps $\langle \pi_3, \ell_3 \rangle$ s.t. $\langle \pi_1, \ell_1 \rangle \rightsquigarrow_{\bar{\ell}} \langle \pi_3, \ell_3 \rangle$ and $\langle \pi_2, \ell_2 \rangle \rightsquigarrow_{\bar{\ell}} \langle \pi_3, \ell_3 \rangle$. The reader can find counter-examples to the strong confluence property in the proofs of Lemma 3.14 and Lemma 3.16.

We thus prove the confluence of ℓ -reduction (Theorem 3.12) by decomposing \rightsquigarrow_ℓ into its logical and structural subreduction, $\rightsquigarrow_{\log\ell}$ and $\rightsquigarrow_{str\ell}$, exactly as we did for \rightsquigarrow_{cut} in Definition 2.11. Then we show that both $\rightsquigarrow_{\log\ell}$ and $\rightsquigarrow_{str\ell}$ are confluent (Proposition 3.15 and Proposition 3.17) and that they commute (Lemma 3.19). We conclude that \rightsquigarrow_ℓ is confluent since it is the union of two confluent reductions which commute (Hindley-Rosen Lemma, see [Klo92], [Py98]).

Definition 3.11 *We define the following subreductions of \rightsquigarrow_ℓ :*

- *the ℓ -logical reduction, denoted by $\rightsquigarrow_{\log\ell}$: a $\rightsquigarrow_{\log\ell}$ -step is one of the ℓ -steps (ax), (\otimes/\wp) , $(1/\perp)$, $(\oplus_i/\&_i)$, $(!/?d)$;*
- *the ℓ -structural reduction, denoted by $\rightsquigarrow_{str\ell}$: a $\rightsquigarrow_{str\ell}$ -step is one of the ℓ -steps $(!/!)$ and $(!/?c)$.*

Of course we have $\rightsquigarrow_\ell = \rightsquigarrow_{\log\ell} \cup \rightsquigarrow_{str\ell}$.

Notice that $\rightsquigarrow_{str\ell}$ is defined precisely by those ℓ -steps which leave unchanged the labels of the $*$ -links.

Theorem 3.12 *The reduction \rightsquigarrow_ℓ is confluent on the s^ℓ ps satisfying AC.*

PROOF. Consequence of Prop. 3.15, Prop. 3.17, Lemma 3.19 and the Hindley-Rosen Lemma [Klo92], [Py98]. \square

In what follows we prove that $\rightsquigarrow_{\log\ell}$ and $\rightsquigarrow_{str\ell}$ are confluent and commute. The difficult part of the proof is already achieved: it consists in establishing that both $\rightsquigarrow_{\log\ell}$ and $\rightsquigarrow_{str\ell}$ are SN, which is an immediate consequence of SN of \rightsquigarrow_{\log} (Proposition 2.14) and of \rightsquigarrow_{str} (Proposition 3.15): the labelling of s^ℓ ps plays no role w.r.t. SN.

Remark 3.13 *One can easily adapt Theorem 3.12 and prove the confluence of \rightsquigarrow_{cut} for \top -free sps satisfying AC: just check that \rightsquigarrow_{\log} and \rightsquigarrow_{str} are locally confluent (i.e. add the erasing steps to the proof of Lemma 3.14 and Lemma 3.16) and that they commute.*

Confluence of $\rightsquigarrow_{\log \ell}$. We prove the confluence of $\rightsquigarrow_{\log \ell}$ (Proposition 3.15) by applying the Newman Lemma ([Klo92],[Py98]): a relation which is locally confluent and SN is confluent. The local confluence of $\rightsquigarrow_{\text{str} \ell}$ is proven by Lemma 3.14, and the SN of $\rightsquigarrow_{\log \ell}$ is an immediate consequence of Proposition 2.14.

Neither in the proof of Proposition 2.14 nor in the one of Lemma 3.14 the AC condition is used, so that confluence of $\rightsquigarrow_{\log \ell}$ is established for the whole set of s^ℓ ps.

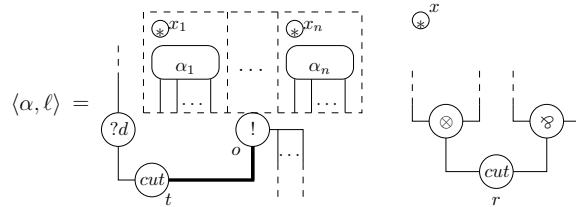
Lemma 3.14 *The reduction $\rightsquigarrow_{\log \ell}$ is locally confluent on the s^ℓ ps.*

PROOF. We prove that for every slice $\langle \alpha, \ell \rangle$ the following diagram holds:

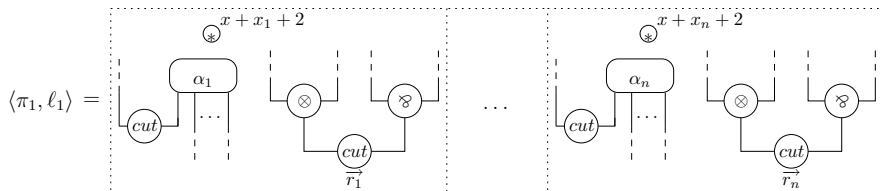
$$\begin{array}{ccc}
 \langle \alpha, \ell \rangle & \rightsquigarrow_{\log \ell} & \langle \pi_2, \ell_2 \rangle \\
 \left. \begin{array}{c} \vdots \\ \log \ell \downarrow \end{array} \right\} & & \left. \begin{array}{c} \vdots \\ \log \ell \downarrow \end{array} \right\} \\
 \langle \pi_1, \ell_1 \rangle & \rightsquigarrow_{\log \ell}^* & \langle \pi_3, \ell_3 \rangle
 \end{array}$$

This immediately entails local confluence for general s^ℓ ps.

Establishing that the previous diagram holds is not immediate only when at least one reduction is of type $(!/?d)$: in this case²¹ the slice $\langle \alpha, \ell \rangle$ is duplicated a number of times equal to the number of slices of the box opened by the $(!/?d)$ reduction. Let us consider for example the case $\langle \alpha, \ell \rangle \rightsquigarrow_{\log} \langle \pi_1, \ell_1 \rangle$ is a $(!/?d)$ step and $\langle \alpha, \ell \rangle \rightsquigarrow_{\log} \langle \pi_2, \ell_2 \rangle$ is a (\otimes/\wp) step (the other cases are similar and left to the reader). We have:

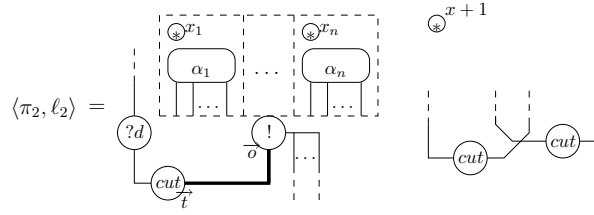


the reduction of the cut t opens the box of the $!$ -link o and duplicates the slice $\langle \alpha, \ell \rangle$ n times, giving as result the following s^ℓ ps (notice that the labelling of the $*$ -links changes):

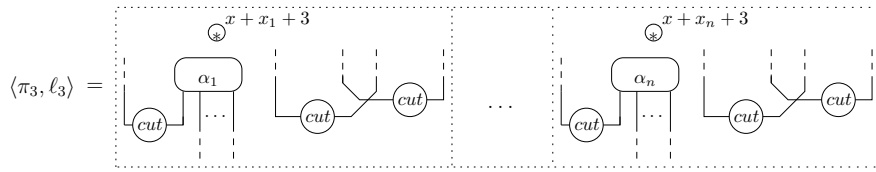


²¹ We can suppose without loss of generality that the reduced cut link has depth 0 in α .

Instead, by reducing the cut r in $\langle \alpha, \ell \rangle$, one obtains the following slice:



If we reduce the cut \vec{t} in $\langle \pi_2, \ell_2 \rangle$, we obtain the following s^ℓps (again notice that the labelling of the *-links changes):



The same s^ℓps can be obtained by reducing the residues $\vec{r}_1, \dots, \vec{r}_n$ of the cut r in $\langle \pi_1, \ell_1 \rangle$. We conclude that $\langle \pi_1, \ell_1 \rangle \rightsquigarrow_{\text{logl}}^* \langle \pi_3, \ell_3 \rangle$ and $\langle \pi_2, \ell_2 \rangle \rightsquigarrow_{\text{logl}} \langle \pi_3, \ell_3 \rangle$. \square

Notice that the critical pair analysed in the proof of Lemma 3.14 is a counter-example to the strong confluence of $\rightsquigarrow_{\text{logl}}$.

Proposition 3.15 *The reduction $\rightsquigarrow_{\text{logl}}$ is confluent on the s^ℓps.*

PROOF. Consequence of Lemma 3.14, Proposition 2.14 and the Newman Lemma [Klo92], [Py98]. \square

Confluence of $\rightsquigarrow_{\text{strl}}$. As for $\rightsquigarrow_{\text{logl}}$, the confluence of $\rightsquigarrow_{\text{strl}}$ (Proposition 3.17) is obtained using the Newman Lemma. The local confluence of $\rightsquigarrow_{\text{strl}}$ is proven by Lemma 3.16, and the SN of $\rightsquigarrow_{\text{strl}}$ is an immediate consequence of Proposition 2.20.

In sharp contrast with the case of the $\rightsquigarrow_{\text{logl}}$ rewriting rule, the reader should remark that the AC condition plays a crucial role both for Lemma 3.16 and Proposition 2.20: in Subsection 2.2 we have given counter-examples (see Fig. 11 and Fig. 12) both to the local confluence and the SN of $\rightsquigarrow_{\text{str}}$ for sps which do not satisfy AC.

Remark that the labelling of the *-links does not play any role in the confluence of $\rightsquigarrow_{\text{strl}}$, since the ℓ function is invariant under $\rightsquigarrow_{\text{strl}}$. We nevertheless picture

explicitly the $*$ -links in the figures illustrating the following lemma, in order to stress that confluence holds for *labelled* sps.

Lemma 3.16 *The reduction $\rightsquigarrow_{str\ell}$ is locally confluent on the s^ℓ ps satisfying AC.*

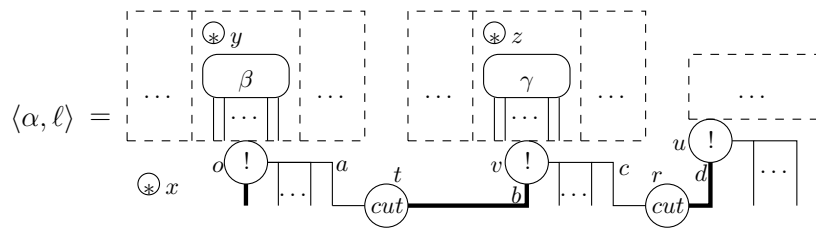
PROOF. The local confluence of $\rightsquigarrow_{str\ell}$ is quite a standard result, the proof is morally the same as that of the local confluence of the exponential reduction in MELL given by V. Danos in its PhD thesis [Dan90]. The local confluence of $\rightsquigarrow_{str\ell}$ can be reduced to the following diagram:

$$\begin{array}{ccc}
 \langle \alpha, \ell \rangle & \rightsquigarrow_{str\ell} & \langle \pi_2, \ell_2 \rangle \\
 \left. \begin{array}{c} \vdots \\ \text{str}\ell \downarrow \end{array} \right\} & & \left. \begin{array}{c} \vdots \\ \text{str}\ell_* \downarrow \end{array} \right\} \\
 \langle \pi_1, \ell_1 \rangle & \rightsquigarrow_{str\ell}^* & \langle \pi_3, \ell_3 \rangle
 \end{array}$$

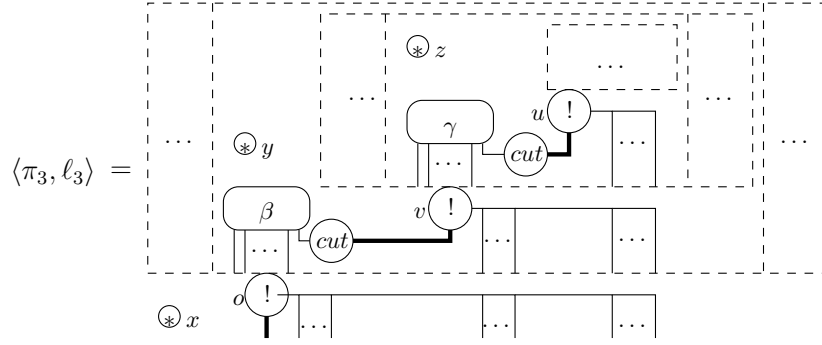
This immediately entails local confluence for general s^ℓ ps.

The proof is by inspection of all possible cases, we treat in detail only the following two critical pairs (the considered cuts have depth 0 in α).

- (1) If $\langle \alpha, \ell \rangle \rightsquigarrow_{str\ell} \langle \pi_1, \ell_1 \rangle$ is the reduction of a cut t of type $(!/!)$ and $\langle \alpha, \ell \rangle \rightsquigarrow_{str} \langle \pi_2, \ell_2 \rangle$ is the reduction of a cut r of type $(!/!)$ and if the two cuts are in opposition as pictured below:

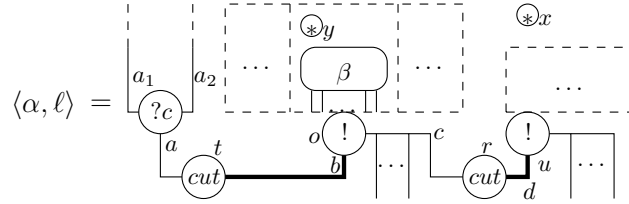


then the reduction of t will bring a copy of v in each slice β of the s^ℓ ps associated with o and it will transform the premise c of the cut r in an auxiliary conclusion of the residue of the $!$ -link o in $\langle \pi_1, \ell_1 \rangle$. On the other hand the reduction of r in $\langle \alpha, \ell \rangle$ will bring a copy of u in each slice γ of the s^ℓ ps associated with v . The s^ℓ ps $\langle \pi_3, \ell_3 \rangle$ is obtained from $\langle \pi_2, \ell_2 \rangle$ by reducing the (unique) residue of t in $\langle \pi_2, \ell_2 \rangle$, which results in bringing v , and hence u , in each slice of the s^ℓ ps associated with o :

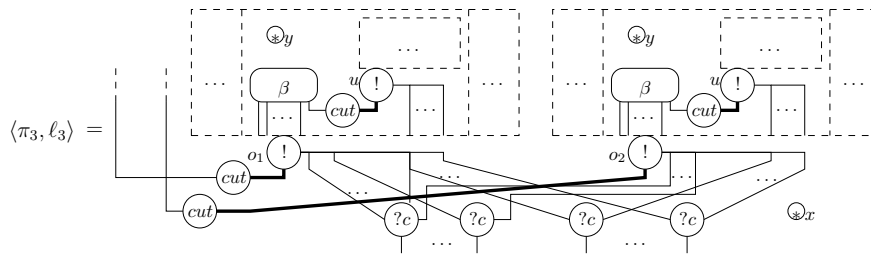


The s^l ps $\langle \pi_3, \ell_3 \rangle$ can also be obtained from $\langle \pi_1, \ell_1 \rangle$ in two steps (recall we are proving only local confluence of $\rightsquigarrow_{str\ell}$): first, we reduce the residue of r in $\langle \pi_1, \ell_1 \rangle$ thus obtaining for each slice β in the box of o a cut between an auxiliary conclusion of a copy of v and a copy of u , and then we reduce every such cut, thus bringing a copy of u in every slice γ in the boxes of the various copies of v , as sketched in the above figure.

- (2) If $\langle \alpha, \ell \rangle \rightsquigarrow_{str\ell} \langle \pi_1, \ell_1 \rangle$ is the reduction of a cut t of type $(!/?c)$ and $\langle \alpha, \ell \rangle \rightsquigarrow_{str\ell} \langle \pi_2, \ell_2 \rangle$ is the reduction of a cut r of type $(!/!)$ and if the two cuts are in opposition as pictured below:



then the reduction of t will duplicate o in $\langle \pi_1, \ell_1 \rangle$ and will transform the cut r in a cut of type $(!/?c)$ between the residue of the $!$ -link u and a created $?c$ -link with premises the auxiliary conclusions corresponding to c of the two copies of o . On the other hand, the reduction of r in $\langle \alpha, \ell \rangle$ will bring a copy of u in each slice β of the s^l ps associated with o . The s^l ps $\langle \pi_3, \ell_3 \rangle$ is obtained from $\langle \pi_2, \ell_2 \rangle$ by reducing the (unique) residue of t in $\langle \pi_2, \ell_2 \rangle$, which duplicates o , and hence u :



The s^l ps $\langle \pi_3, \ell_3 \rangle$ can also be obtained from $\langle \pi_1, \ell_1 \rangle$ in three steps (also in this case we have only local confluence and not strong confluence): first, we

reduce the residue of r in $\langle \pi_2, \ell_2 \rangle$, which is a cut of type $(!/?c)$; this duplicates u and creates two cuts of type $(!/!)$, one between the first copies of o and u and the other one between the second copies of o and u . We obtain $\langle \pi_3, \ell_3 \rangle$ by further reducing the two $(!/!)$ cuts.

□

The critical pairs treated in the proof of Lemma 3.16 are counter-examples to the strong confluence of $\rightsquigarrow_{str\ell}$.

Proposition 3.17 *The reduction $\rightsquigarrow_{str\ell}$ is confluent on the s^ℓ ps satisfying AC.*

PROOF. Consequence of Lemma 3.16, Proposition 2.20 and the Newman Lemma [Klo92], [Py98]. □

Commutation of $\rightsquigarrow_{log\ell}$ and $\rightsquigarrow_{str\ell}$. The last step allows to merge $\rightsquigarrow_{log\ell}$ and $\rightsquigarrow_{str\ell}$ together, proving that the two reductions commute (Lemma 3.19): this is achieved by applying a lemma by Di Cosmo, Piperno and Geser ([Py98]) which reduces the commutation of $\rightsquigarrow_{log\ell}$ and $\rightsquigarrow_{str\ell}$ to the diagram pictured in Lemma 3.18.

Like in the proof of confluence for $\rightsquigarrow_{log\ell}$ (and contrary to the proof of confluence for $\rightsquigarrow_{str\ell}$) the AC condition is not needed in this paragraph.

Lemma 3.18 *For every s^ℓ ps the following diagram holds:*

$$\begin{array}{ccc}
 \langle \pi, \ell \rangle & \rightsquigarrow_{log\ell} & \langle \pi_2, \ell_2 \rangle \\
 \downarrow \scriptstyle{str\ell} & & \downarrow \scriptstyle{str\ell}^* \\
 \langle \pi_1, \ell_1 \rangle & \rightsquigarrow_{log\ell}^+ & \langle \pi_3, \ell_3 \rangle
 \end{array}$$

PROOF. Also for this lemma we restrict ourselves to the case $\langle \pi, \ell \rangle$ is a slice $\langle \alpha, \ell \rangle$: the general case follows immediately.

Let t (resp. r) be the cut link of α reduced in $\langle \alpha, \ell \rangle \rightsquigarrow_{str\ell} \langle \pi_1, \ell_1 \rangle$ (resp. in $\langle \alpha, \ell \rangle \rightsquigarrow_{log\ell} \langle \pi_2, \ell_2 \rangle$). The proof is by induction on the depth of r . We split the proof in three cases.

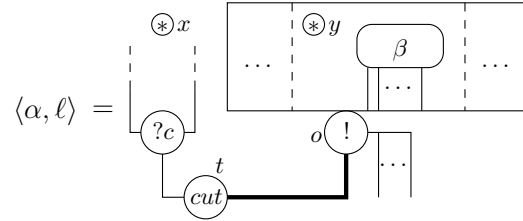
Case (i). The cut r is at depth 0 in $\langle \alpha, \ell \rangle$: this case is immediate, since the reduction of t does not affect r . The only slightly delicate case is when r is of type $(!/?d)$ (so that its reduction duplicates $\langle \alpha, \ell \rangle$) and t is duplicated a number of times equal to the number of slices, say n , of the opened exponential box: $\pi_2 = [\alpha_1, \dots, \alpha_n]$. In particular, let $\vec{t}_1, \dots, \vec{t}_n$ be the n residues of t in $\langle \pi_2, \ell_2 \rangle$. We have two subcases:

(i) if one of the two premises of t is an auxiliary conclusion of the $!$ -link opened by the reduction of r , then we do not need to reduce the residues of t to close the diagram: $\langle \pi_3, \ell_3 \rangle = \langle \pi_2, \ell_2 \rangle$ ²²; (ii) otherwise the reduction of the residues of t in $\langle \pi_2, \ell_2 \rangle$ gives as result the same s^ℓ ps as the one given by the reduction of the (unique) residue of r in $\langle \pi_1, \ell_1 \rangle$.

Case (ii). Both t and r are cuts in boxes of α . Let o (resp. u) be the $!$ -link at depth 0 of α containing t (resp. r). If o and u are different $!$ -links, then this case is immediate; if they are the same $!$ -link o , let $\langle \pi^o, \ell^o \rangle$ be the box associated with o . We have $\langle \pi^o, \ell^o \rangle \rightsquigarrow_{str\ell} \langle \pi_1^o, \ell_1^o \rangle$ and $\langle \pi^o, \ell^o \rangle \rightsquigarrow_{log\ell} \langle \pi_2^o, \ell_2^o \rangle$. We apply the induction hypothesis and we obtain an s^ℓ ps $\langle \pi_3^o, \ell_3^o \rangle$ s.t. $\langle \pi_1^o, \ell_1^o \rangle \rightsquigarrow_{log\ell}^+ \langle \pi_3^o, \ell_3^o \rangle$ and $\langle \pi_2^o, \ell_2^o \rangle \rightsquigarrow_{str\ell}^* \langle \pi_3^o, \ell_3^o \rangle$. The s^ℓ ps $\langle \pi_3, \ell_3 \rangle$ is then obtained from $\langle \pi, \ell \rangle$ by associating with o the s^ℓ ps $\langle \pi_3^o, \ell_3^o \rangle$.

Case (iii). The cut t is at depth 0 and r is contained in the $!$ -link o at depth 0 of α : then we can have two critical pairs, since the $!$ -link o may be involved in the reduction of t .

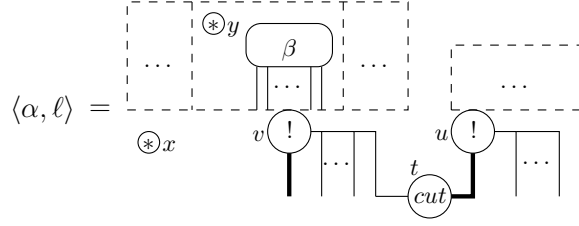
One critical pair is when t is of type $(!/?c)$ and o is the $!$ -link which is duplicated by the reduction of t :



The reduction of t duplicates o , hence r , in $\langle \pi_1, \ell_1 \rangle$, but does not change the content of the box nor the labels of the $*$ -links. On the other hand, the reduction of r changes the s^ℓ ps associated with o , without affecting the cut t nor the label of the $*$ -link at depth 0 of α , but possibly changing the labels of the $*$ -links inside o . The reader can check that reducing the two residues of r in $\langle \pi_1, \ell_1 \rangle$ gives the same result $\langle \pi_3, \ell_3 \rangle$ as reducing the residue of t in $\langle \pi_2, \ell_2 \rangle$. By the way notice that the reduction $\langle \pi_1, \ell_1 \rangle \rightsquigarrow_{log\ell}^+ \langle \pi_3, \ell_3 \rangle$ costs two steps, due to the duplication of r .

The other critical pair is when t is of type $(!/!)$ and o is one of the two $!$ -links involved in the reduction of t (i.e. in the figure below $o = u$ or $o = v$):

²² Notice that in this case (and only in this case) these t 's residues might also be non reducible cuts.



The reduction of t brings one copy of u in each slice of the box associated with v (notice that v contains at least one slice, since $\rightsquigarrow_{str\ell}$ is a non-erasing reduction step): however, this reduction does not change the s^ℓ ps associated with u nor the labels of the $*$ -links. On the other hand the reduction of r in $\langle \alpha, \ell \rangle$ changes the s^ℓ ps associated with u or that associated with v , depending on whether $o = u$ or $o = v$.

If $o = u$, then the case is simple: let $\vec{r}_1, \dots, \vec{r}_n$ (where $n \geq 1$, as we noticed above) be the residues of r in $\langle \pi_1, \ell_1 \rangle$. Reducing these n cuts gives the same result as reducing the unique residue of t in $\langle \pi_2, \ell_2 \rangle$.

If $o = v$, then one has to notice that even if the reduction of t changes the box π^v associated with v (bringing “inside v ” the !-link u), this reduction does not affect any residue of any cut of any slice of π^v . In particular we can reduce r ’s residue in $\langle \pi_1, \ell_1 \rangle$ like the reduction step $\langle \alpha, \ell \rangle \rightsquigarrow_{log\ell} \langle \pi_2, \ell_2 \rangle$ does, thus obtaining the s^ℓ ps $\langle \pi_3, \ell_3 \rangle$ which is also the result of reducing the residue of t in $\langle \pi_2, \ell_2 \rangle$. \square

Lemma 3.19 *The reductions $\rightsquigarrow_{str\ell}$ and $\rightsquigarrow_{log\ell}$ commute, i.e.:*

$$\begin{array}{ccc} \langle \pi, \ell \rangle & \rightsquigarrow_{log\ell}^* & \langle \pi_2, \ell_2 \rangle \\ \downarrow \text{str}\ell_* & & \downarrow \text{str}\ell_* \\ \langle \pi_1, \ell_1 \rangle & \rightsquigarrow_{log\ell}^* & \langle \pi_3, \ell_3 \rangle \end{array}$$

PROOF. It is a consequence of Lemma 3.18, Prop. 2.14 and a Lemma by Di Cosmo, Piperno and Geser ([Py98]): let $\rightsquigarrow_1, \rightsquigarrow_2$ be two reductions s.t. \rightsquigarrow_1 is SN and the following diagram holds

$$\begin{array}{ccc} \pi & \rightsquigarrow_1 & \langle \pi_2 \rangle \\ \downarrow 2 & & \downarrow 2^* \\ \langle \pi_1 \rangle & \rightsquigarrow_2^+ & \langle \pi_3 \rangle \end{array}$$

then \rightsquigarrow_1 and \rightsquigarrow_2 commute. \square

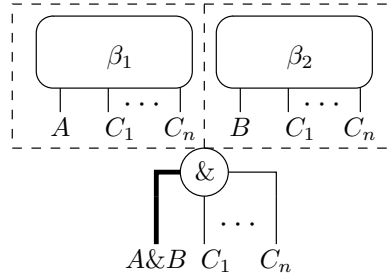


Fig. 18. An example of additive box

4 Strong Normalization for Linear Logic

We now want to apply our (rather general) main result (Theorem 3.2) in order to prove strong normalization for full second order LL (Theorem 4.10). As proof-nets, we take here the more general currently used notion, obtained by combining [Gir87a], [Gir91] and [Dan90], and by generalizing the $(ccad)$ and the $(\top - cc)$ reduction steps (see below). These proof-nets are defined in [Tdf03]: let's call them simply *nets*, and denote them with initial Greek letters $\alpha, \beta \dots$

This last section requires some knowledge on nets, mainly some acquaintance with the additive and second order cut-elimination and the notion of sequentialization: all the precise definitions are in [Tdf03].

4.1 The syntax of nets

There are two main differences between nets and sps: (i) nets are typed by second order LL formulas (Figure 17 recalls the grammar of LL) and (ii) nets use additive boxes to handle the $\&$ rule. Let us comment this last feature.

$F := a$	$ a^\perp$
1	$ \perp$
$F \otimes F$	$ F \wp F$
0	$ \top$
$F \oplus F$	$ F \& F$
$!F$	$?F$
$\forall a.F$	$ \exists a.F$

Fig. 17. Grammar of LL formulas

In the framework of nets, $\&$ -links behave like $!$ -links (see Figure 18): they have 0 premises and $n + 1$ conclusions, a distinguished one (the main conclusion, of type, say, $A \& B$) and possibly others (the auxiliary conclusions); with every $\&$ -link with $n + 1$ conclusions are associated two nets β_1 and β_2 , called *first (or left)* and *second (or right)* component of the $\&$ -link. The nets β_1 and β_2 have the same n conclusions as the auxiliary conclusions of the $\&$ -link (called the auxiliary conclusions of the component of the $\&$ -link) and one distinguished conclusion (of type respectively A and B , and called the main conclusion of the component of the $\&$ -link).

The presence of additive boxes has two main conse-

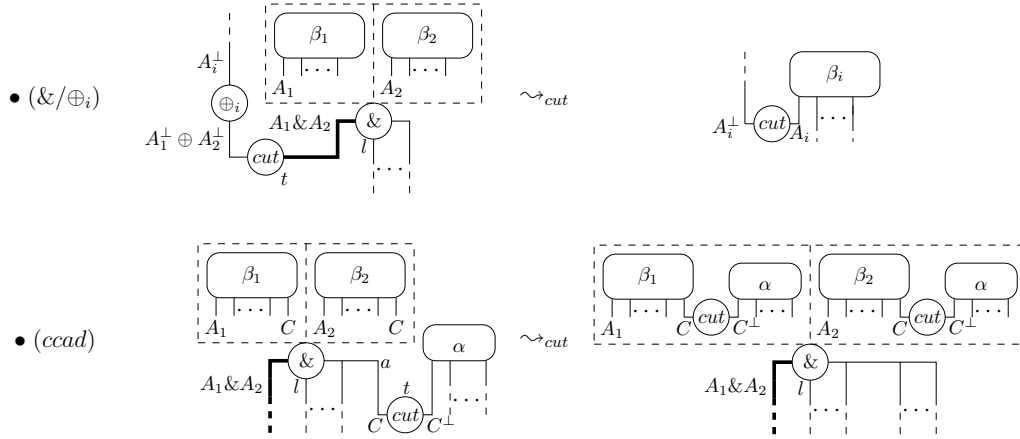


Fig. 19. The cut-elimination steps associated with the additive box

quences in the definition of the cut-elimination steps: (1) there is a unique additive step ($\&/\oplus_i$) which combines an erasing feature (like the step ($\&_i/\oplus_j$) with $i \neq j$, $i, j \in \{1, 2\}$ in sps) and a non-erasing feature (like the step ($\&_i/\oplus_i$) in sps); (2) it yields a new type of cut-link (called (*ccad*)), which is the nightmare of this way of representing proofs (the same holds for the ($\top - cc$) cut of Definition 2.6).

The step ($\&/\oplus_i$). A cut t of a net β is of type ($\&/\oplus_i$) if one premise is the main conclusion of a $\&$ -link and the other premise is conclusion of a \oplus_i -link (see Figure 19). We reduce t by erasing the \oplus_i -link, by substituting the $\&$ -link with its i^{th} component and cutting the premise of the erased \oplus_i -link and the main conclusion of the i^{th} component of the $\&$ -link. As already mentioned, this reduction step has both an erasing nature (it erases one component of the $\&$ -link) and a non-erasing nature (it modifies the cut formula like for example the (\otimes/\wp) step). This mix is critical with respect to standardization, as we will discuss in details in the next Subsection 4.2.

The step (*ccad*). A cut t of a net β is of type (*ccad*) if one of its premises, say a , is an auxiliary conclusion of a $\&$ -link l (see Figure 19). We should reduce t by selecting a subnet α of β (not containing l) having the premise of t different from a among its conclusions. We then substitute l , the cut link t and α by a new $\&$ -link (which we still call l), having the same conclusions as the original l where we have substituted the edge a by the conclusions of α (different from t 's premise). The i^{th} component of the new link l is obtained by cutting the conclusion corresponding to the edge a of the i^{th} component of the original l and the conclusion of α which is a premise of t . The point is that it is not clear which subnet α should be selected (for example, in [Gir87a] α is the *empire of a*, in [TdF03] α is any subnet having a among its conclusions). We take here the (more general) option of [TdF03], which is also applied to the ($\top - cc$) step. Notice that, as for ($\top - cc$) (Remark 2.7), the presence of (*ccad*)-reduction steps entails the failure of confluence.

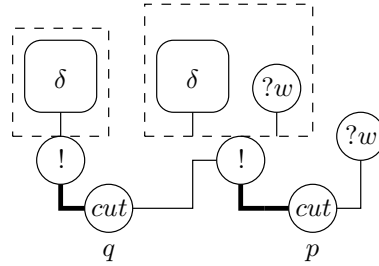


Fig. 20. An untyped MELL net which is WN but not SN (δ is the slice of Figure 8)

Despite these differences, the cut-elimination procedure for sps is the key tool to prove SN for nets. We associate (Subsection 4.3) with every net β its set of slices $\text{sl}(\beta)$ (which is an sps), and we prove that $\beta \in \text{SN}$ by proving that $\text{sl}(\beta) \in \text{SN}$. More precisely, we prove that for every net β , (i) $\text{sl}(\beta)$ satisfies AC, (ii) $\beta \in \text{SN}$ if $\text{sl}(\beta) \in \text{SN}$, (iii) $\text{sl}(\beta) \in \text{SN}$ if $\text{sl}(\beta) \in \text{WN}^{-e}$, (iv) $\text{sl}(\beta) \in \text{WN}^{-e}$. Point (i) is trivial (Proposition 4.1); point (ii) is Proposition 4.2; point (iii) is a consequence of point (i) and of our standardization theorem (Theorem 3.2), morally this is the real missing point in Girard’s original proof; point (iv) is the “difficult” part of the SN proof (which is outside Peano arithmetic): one uses Girard’s reducibility candidates to prove $\text{sl}(\beta) \in \text{WN}^{-e}$. It is well known that this kind of tool is very powerful and can be adapted to a lot of different situations. Indeed, Girard’s proof of [Gir87a] works perfectly well if one substitutes “strong normalization” with “weak normalization”, and the reader acquainted with reducibility candidates is probably already convinced that the changes needed to prove $\text{sl}(\beta) \in \text{WN}^{-e}$ (instead of $\beta \in \text{WN}$) present no major difficulty. We nevertheless give the precise definition of reducibility candidate (Definition 4.8) that we need in order to prove Theorem 4.9 (that is point (iv)), thus concluding with the strong normalization theorem for second order LL nets (Theorem 4.10).

Apparently, an alternative solution was to use Theorem 3.2 in order to prove the original Girard’s standardization theorem (Theorem 4.25 p.72 of [Gir87a]). We explain in the next Subsection 4.2 why this alternative fails.

4.2 A digression on standardization

Before proving the points (i)-(iv) stated above, let us discuss the alternative solution mentioned, namely to use Theorem 3.2 in order to prove the original standardization theorem (Theorem 4.25 p.72 of [Gir87a]). We claim this approach fails, since the original standardization is based on a “wrong” definition of standard reduction. To help the reader, we reproduce verbatim Definition 4.24 and Theorem 4.25 p.72 of [Gir87a].

4.24. Definition. A contraction is *standard* when it does not erase any symbol CUT besides the one explicitly considered. Concretely, this means that some parts of the configuration we replace have to be cut-free, namely:

- in $(\&/\oplus_1 - SC)$, β_2 must be cut-free,
- in $(\&/\oplus_2 - SC)$, β_1 must be cut-free,
- in $(!/?w - SC)$, β_1 must be cut-free,
- in $(\top - SC)$, eA^\perp must be cut-free.

A reduction is standard when made of standard contractions.

4.25 Theorem (Standardization Lemma). *Let β be a proof-net and assume that there is a standard reduction from β to a cut-free β' . Then β is SN.*

The problem with the above definition of standard reduction (*contraction* in the language of [Gir87a]) was already pointed out in [Dan90] in the restricted MELL framework: recall the slice $\delta\delta$ of Figure 9, which can be considered an “untyped MELL net”²³. As we pictured in Figure 10, we have $\delta\delta \rightsquigarrow_{cut}^+ \delta\delta$. Now consider the untyped net β of Figure 20. By reducing the cut p of β one obtains a strongly normalizing untyped net; however β is *not* strongly normalizing (reduce the cut q and you will get a net having the net $\delta\delta$ of Figure 9 as subnet), despite the fact that the reduction step associated with p is standard in Girard’s sense (Definition 4.24 p.72 of [Gir87a]). From our point of view, this means that Theorem 4.25 p.72 of [Gir87a] is “morally wrong”, even though it is “technically correct” (it deals only with typed nets, so its conclusion is true...). The solution proposed in [Dan90] was to freeze the cut link p and consider “strict” reductions, that is the rewriting rule obtained by forbidding $(!/?w)$ reduction steps: this leads to théorème 8.31 p.64 of [Dan90], allowing to prove SN for MELL nets.

In presence of the additives, the same phenomenon occurs, but the solution cannot be that simple, as we now explain. Consider the untyped net β of Figure 21. Again, by reducing the cut p one obtains a strongly normalizing untyped net; however β is *not* strongly normalizing (reduce the cut q and the $(ccad)$ created by the reduction of q), despite the fact that the reduction step associated with p is standard in Girard’s sense. Following Danos, one would be tempted to freeze p , but this wouldn’t be correct. Indeed, we should then freeze all the cuts of type $(\&/\oplus_i)$, and we see very well that such a freezing hides infinite reduction sequences: the untyped net β would be in normal form w.r.t. the rewriting rule induced by this freezing, despite the fact that $\beta \notin SN$. The difference between the reduction steps $(!/?w)$ and $(\&/\oplus_i)$ is that while the first one is purely erasing, the second one is both erasing and non erasing. A key feature of the introduction of slices is precisely to separate the erasing aspect of the $(\&/\oplus_i)$ reduction step (taken into account by the $(\&/\oplus_i)$

²³ In this short discussion, we use the expression “untyped net”, referring to the untyped version of the nets considered in this Subsection 4.1. An untyped net is not necessarily an sps because of the presence of additive boxes.

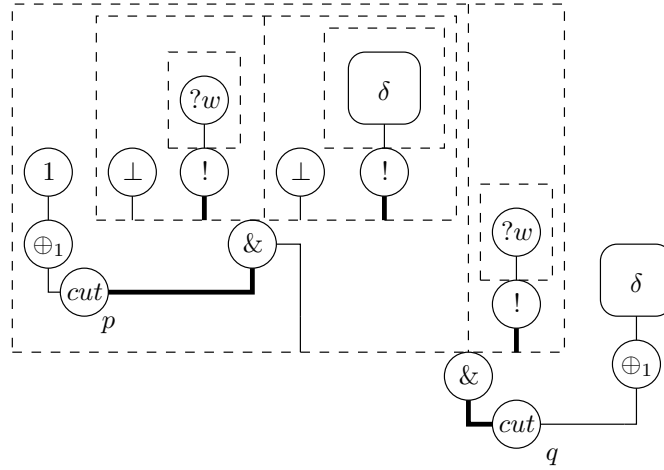


Fig. 21. An untyped LL net which is WN but not SN (δ is the slice of Figure 8)

steps with $i \neq j$) from its non erasing aspect (taken into account by the $(\&_i/\oplus_i)$ steps).

For this reason the standardization theorem for full LL can be correctly stated only for sliced pure structures (as we did in Theorem 3.2) and not for nets.

Another attempt to prove SN for LL (in presence of the additives) is contained in Okada's work ([Oka99]). The method proposed is to use phase semantics, which works well for WN, but for SN the same problem as in [Gir87a] arises (the proof of the standardization theorem), and we have to mention here that (like for [Gir87a]) the way Okada argues on this point cannot be considered convincing (the author himself uses the expression "Sketch of proof"). Indeed:

- (1) Okada claims that (in certain circumstances) if $\beta \rightsquigarrow_{cut} \beta'$ and the reduced cut is of type $(!/!)$ and if $\beta' \in \text{SN}$, then $\beta \in \text{SN}$ ("Sketch of Proof" of lemma 6.6 p.364 case (4) of [Oka99]). Of course (like Theorem 4.25 p.72 of [Gir87a]) the statement above is true since eventually all nets turn out to be strongly normalizable. However, the author claims that it is possible to "simulate" any reduction sequence of β by a reduction sequence of β' . This is as difficult as proving standardization, and it is the motivation for théorème 8.31 p.64 of [Dan90]: in order to solve the problem Danos had to make the first very sharp analysis of the rewriting rule induced by LL cut-elimination (in particular he had to prove confluence for MELL);
- (2) in presence of the additives a *very* "restricted" cut-elimination procedure is defined (see the "&-box entering rule" of p.373 of [Oka99]), so that even in case the proof were considered convincing (and we believe it shouldn't be) it would still be incomplete. With respect to this restricted procedure the extension to the additives is analysed ("Sketch of Modified Proof of lemma 6.6" p.379 of [Oka99]). Actually, Okada's restriction is a way to eliminate the

intrinsic difficulty of the (*ccad*) reduction step: in this paper we found a way to keep control on this step in presence of all the other LL connectives.

4.3 Slicing nets

Let us come back to the proof of SN for nets. We hinted in Subsection 4.1 that the SN of a net is related to the SN of the sps which is “behind” that net. Indeed, a net β can be *sliced* in an sps $\text{sl}(\beta)$ which has the same number of conclusions as β (an example is given in Figure 22). Following [Gir87a] we define $\text{sl}(\beta)$ by induction on a sequentialization of β :²⁴

- if β is an ax -link (resp. 1-link, \top -link), then $\text{sl}(\beta)$ has only one slice, consisting of an ax -link (resp. 1-link, \top -link);
- if β is a $\&$ -link l s.t. with l is associated the left (resp. right) component β' (resp. β''), then $\text{sl}(\beta)$ is obtained by adding a $\&_1$ -link (resp. $\&_2$ -link) to every slice of $\text{sl}(\beta')$ (resp. $\text{sl}(\beta'')$) and by taking the union of these multisets of slices;
- if β is a $!$ -link l s.t. with l is associated a net β' , then $\text{sl}(\beta)$ has only one slice consisting of a $!$ -link corresponding to l and s.t. with that link is associated $\text{sl}(\beta')$;
- if a conclusion of β is conclusion of a \wp -link (resp. of a \perp -, \oplus_i -, $?w$ -, $?d$ -, $?c$ -link) l , let β' be the subnet of β obtained by erasing l (and its conclusion), then $\text{sl}(\beta)$ is obtained by adding to every slice of $\text{sl}(\beta')$ the \wp -link (resp. \perp -, \oplus_i -, $?w$ -, $?d$ -, $?c$ -link) corresponding to l ;
- if a conclusion of β is conclusion of a \forall - or \exists -link l , let β' be the subnet of β obtained by erasing l (and its conclusion), then $\text{sl}(\beta) = \text{sl}(\beta')$;
- if a conclusion of β is conclusion of a \otimes -link (resp. cut) l s.t. β is the union of two disjoint subnets β' , β'' and of l , then $\text{sl}(\beta)$ is obtained by connecting every slice of $\text{sl}(\beta')$ and every slice of $\text{sl}(\beta'')$ by means of the \otimes -link (resp. cut) corresponding to l .

We first note that the slicing of a net is switching acyclic:

Proposition 4.1 *If β is a net, then $\text{sl}(\beta)$ satisfies AC.*

PROOF. By the correctness criterion for nets (see [Gir87a],[Tdf03]) and the definition of sl . \square

²⁴ The procedure of *slicing* is actually independent from the chosen sequentialization and it can be applied also to non-sequentializable proof-structures (see [Gir96]).

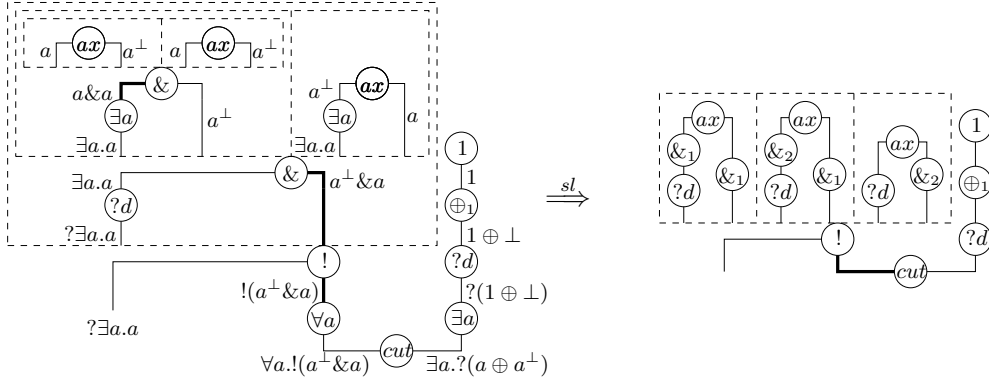


Fig. 22. An example of slicing a net

Then we turn to the crucial point: the cut-elimination of nets can be simulated by that of its slicing, i.e.

$$\begin{array}{ccc}
 \beta & \rightsquigarrow_{cut} & \beta' \\
 \Downarrow_{sl} & & \Downarrow_{sl} \\
 sl(\beta) & \rightsquigarrow_{cut}^* & sl(\beta')
 \end{array}$$

An example of this simulation is given in Figure 23. The reader should notice that a sequence of steps of type $(ccad)$ or (\forall/\exists) (denoted by $\rightsquigarrow_{(ccad),(\forall/\exists)}^*$) is “invisible” in sps: this might be a problem to derive $\beta \in \text{SN}$ from $sl(\beta) \in \text{SN}$, but actually it isn’t thanks to Lemma 4.4. We thus have:

Proposition 4.2 *Let β be a net. If $sl(\beta) \in \text{SN}$ then $\beta \in \text{SN}$.*

PROOF. Immediate consequence of lemmata 4.3 and 4.4. □

Lemma 4.3 (Simulation) *Let β , β' and β'' be three nets, let x be a cut link of β' s.t. $\beta' \rightsquigarrow_x \beta''$ is not a $(ccad)$ nor a (\forall/\exists) reduction step. If $\beta \rightsquigarrow_{(ccad),(\forall/\exists)}^* \beta' \rightsquigarrow_x \beta''$, then $sl(\beta) = sl(\beta') \rightsquigarrow^+ sl(\beta'')$.*

Lemma 4.4 *There is no infinite sequence of reduction steps of type $(ccad)$ or (\forall/\exists) starting from a net.*

PROOF. We use here (a straightforward variant of) the separation property proven in [Tdf03]: if $\beta \rightsquigarrow_{(ccad),(\forall/\exists)}^* \beta'$ and l'_1, l'_2 are two residues²⁵ in β' of a $\&$ -link l of β , then there exists a $\&$ -link l' of β' which “separates” l'_1 and l'_2 , meaning that

²⁵ We refer here to the obvious adaptation to nets of the notion introduced in Definition 2.8 for sps.

l'_1 (resp. l'_2) is a link of the component i (resp. j) of l' and $i \neq j$ (in particular, l'_1 cannot be a link of a component of l'_2 , nor the converse).

Let us define the additive depth of a link l of a net β as the number of $\&$ -components of β containing l , and the additive depth of β as the maximal additive depth of its links.

The separation property implies that for every net β which contains n $\&$ -links, if $\beta \rightsquigarrow^*_{(ccad), (\forall/\exists)} \beta'$, then the additive depth of β' must be less or equal than n . It is then easy to define a size on every $(ccad)$, (\forall/\exists) -reduct of β which shrinks at every $(ccad)$ or (\forall/\exists) step: consider the n -uple whose i^{th} component is the number of links of the net having additive depth i , and order these n -uples lexicographically. \square

4.4 Strong normalization for nets

Finally the last point: we prove for every net β that $\text{sl}(\beta) \in \text{WN}^{-e}$ (Theorem 4.9). Here we need Girard's reducibility candidates. We give the particular definition of reducibility candidates (Definition 4.8) required to prove Theorem 4.9, and then we just sketch the proof of the theorem, being standard after [Gir87a].

Definition 4.5 *A term of type A is a net β together with a distinguished conclusion which is labelled by A . If β (resp. β') is a term of type A (resp. A^\perp), we denote by $\text{CUT}(\beta, \beta')$ the net obtained by connecting β and β' by means of a cut with premises the two distinguished conclusions A, A^\perp .*

Definition 4.6 (duality) *Let X be a set of terms of type A ; we define X^\perp as follows:*

$$X^\perp = \{ \beta' \text{ s.t. } \beta' \text{ term of type } A^\perp \text{ and } \text{sl}(\text{CUT}(\beta, \beta')) \in \text{WN}^{-e} \text{ for every } \beta \in X \}$$

Proposition 4.7 *Let X be a set of terms of type A :*

- (1) *if X contains the axiom link with conclusion A, A^\perp , then for every $\beta \in X^\perp$ one has $\text{sl}(\beta) \in \text{WN}^{-e}$;*
- (2) *if for every $\beta \in X$ one has $\text{sl}(\beta) \in \text{WN}^{-e}$, then X^\perp contains the axiom link with conclusion A, A^\perp .*

PROOF. Immediate by the definitions. \square

Definition 4.8 (reducibility candidate) *A reducibility candidate of type A is a set X of terms of type A s.t.:*

- (1) $X \neq \emptyset$;
- (2) *for every $\beta \in X$, one has $\text{sl}(\beta) \in \text{WN}^{-e}$;*

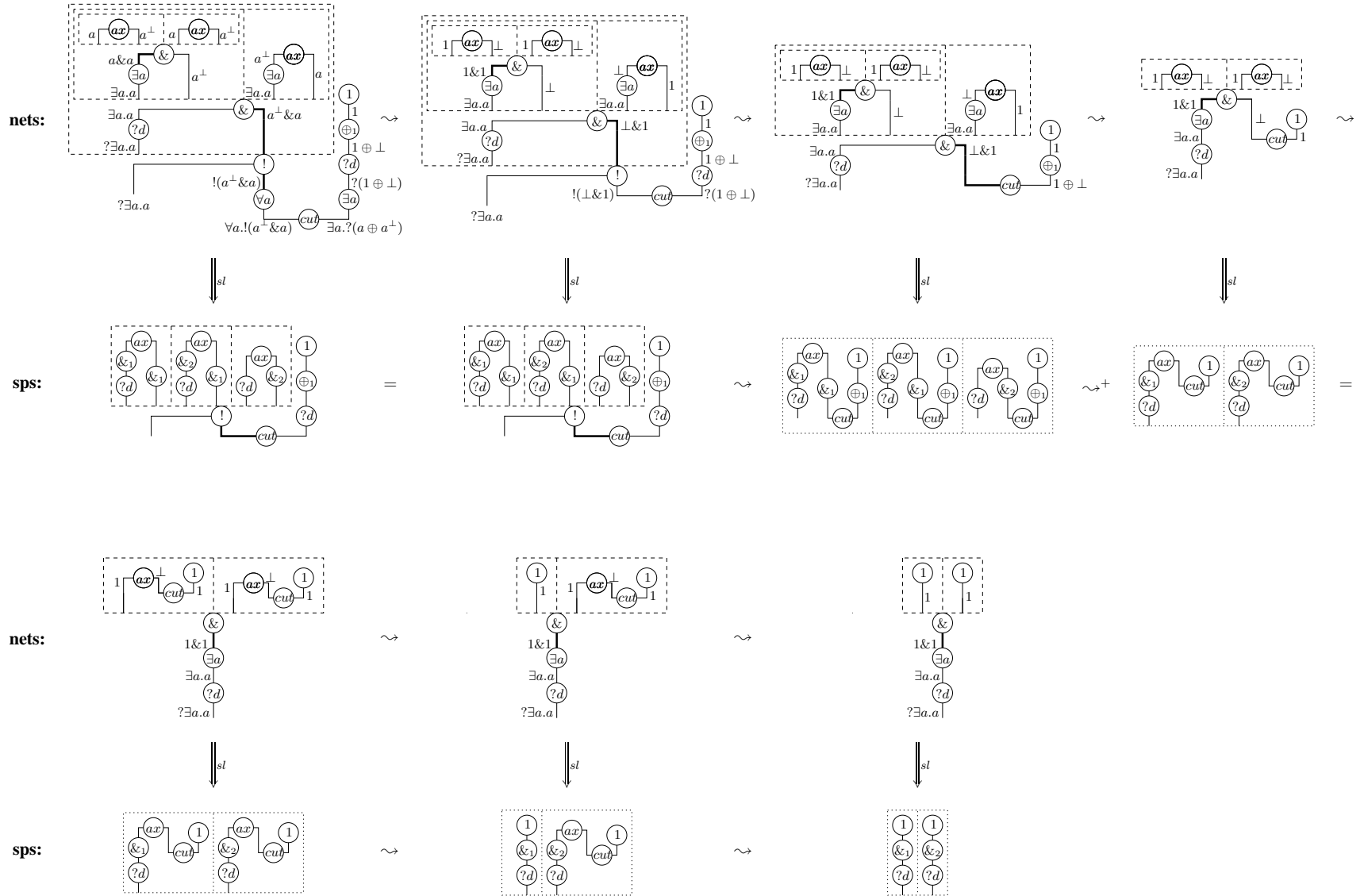


Fig. 23. Example of how sps cut-elimination simulates that of nets: notice that the (*ccad*) and (\forall/\exists) steps are invisible in sps reduction

(3) $X = X^{\perp\perp}$.

Theorem 4.9 (WN^{-e} theorem) *If β is a net, then $\text{sl}(\beta) \in \text{WN}^{-e}$.*

PROOF. Girard's proof of Theorem 4.26 ([Gir87a]) works perfectly if one substitutes " β is SN" with " $\text{sl}(\beta) \in \text{WN}^{-e}$ ". We just give here the idea of how to adapt the original proof. In what follows we use the notations and definitions of [Gir87a]. Let β be a net with conclusions \mathbf{C} (where $\mathbf{C} = C_1, \dots, C_n$ is a sequence of LL formulas), we prove that β is *reducible*, which means that the following holds (see Definition 4.26.8 of [Gir87a]): let \mathbf{a} ($= a_1, \dots, a_m$) be the list of all free variables of \mathbf{C} , then for every sequence of formulas \mathbf{B} ($= B_1, \dots, B_m$), every sequence \mathbf{X} ($= X_1, \dots, X_m$) of reducibility candidates of types \mathbf{B} and every sequence of terms \mathbf{t} ($= t_1, \dots, t_n$) in $\text{RED}(\mathbf{C}^\perp[\mathbf{X}/\mathbf{a}])$ (see Definition 4.26.6 of [Gir87a]), one has $\text{sl}(\text{CUT}(\beta[\mathbf{B}/\mathbf{a}]; \mathbf{t})) \in \text{WN}^{-e}$.

The proof is by induction on a sequentialization of β , hence it splits in 16 cases (the number of rules of LL: $ax, cut, \otimes, 1, \wp, \perp, \&, \top, \oplus_1, \oplus_2, !, ?w, ?d, ?c, \forall^2, \exists^2$). We consider only two cases: the $\&$ - and $!$ -cases, which show how works our version of reducibility candidates.

$\&$ -case: β is obtained from β_1 and β_2 by the $\&$ -box in Figure 18. After simplification, we see that we have to check that $\text{sl}(\text{CUT}(\beta; \mathbf{c}, \oplus_1 t)) \in \text{WN}^{-e}$ and $\text{sl}(\text{CUT}(\beta; \mathbf{c}, \oplus_2 u)) \in \text{WN}^{-e}$ for any $\mathbf{c} \in \text{RED}(\mathbf{C})^\perp$, $t \in \text{RED}(A)^\perp$, and $u \in \text{RED}(B)^\perp$. By induction hypothesis $\text{sl}(\text{CUT}(\beta_1; \mathbf{c}, t)) \in \text{WN}^{-e}$ and $\text{sl}(\text{CUT}(\beta_2; \mathbf{c}, u)) \in \text{WN}^{-e}$. In particular, by Proposition 4.7, we deduce that $\text{sl}(\text{CUT}(\beta_2; \mathbf{c})) \in \text{WN}^{-e}$ (since u can be chosen as an axiom) and $\text{sl}(\oplus_1 t) \in \text{WN}^{-e}$ (since the slicing of every element of a reducibility candidate is WN^{-e}). Consider now $\text{sl}(\text{CUT}(\beta; \mathbf{c}, \oplus_1 t)) = \sigma_1 + \sigma_2$, where σ_1 (resp. σ_2) is the multiset of the slices of $\text{CUT}(\beta; \mathbf{c}, \oplus_1 t)$ which contain the component β_1 (resp. β_2) of the $\&$ -box. Notice that $\sigma_1 \rightsquigarrow_{(\&_1/\oplus_1)} \text{sl}(\text{CUT}(\beta_1; \mathbf{c}, t))$, hence $\sigma_1 \in \text{WN}^{-e}$. But pay attention that we cannot yet infer that also $\sigma \in \text{WN}^{-e}$, because for reducing σ to $\text{sl}(\text{CUT}(\beta_1; \mathbf{c}, t))$ we should perform several $(\&_2/\oplus_1)$ reductions, which are erasing. Nevertheless we deduce that σ_2 is WN^{-e} since $\text{sl}(\text{CUT}(\beta_2; \mathbf{c})) \in \text{WN}^{-e}$ and $\text{sl}(\oplus_1 t) \in \text{WN}^{-e}$ and σ_2 is obtained by connecting every slice of $\text{sl}(\text{CUT}(\beta_2; \mathbf{c}))$ and every slice of $\text{sl}(\oplus_1 t)$ by means of a $(\&_2/\oplus_1)$ cut link. We conclude that $\text{sl}(\text{CUT}(\beta; \mathbf{c}, \oplus_1 t)) \in \text{WN}^{-e}$.

For symmetric reasons, $\text{sl}(\text{CUT}(\beta; \mathbf{c}, \oplus_2 u)) \in \text{WN}^{-e}$.

$!$ -case: β is a $!$ -link with conclusions $!A, ?C_1, \dots, ?C_n$ whose associated box is the net β' . The induction hypothesis is that $\text{sl}(\text{CUT}(\beta'; \mathbf{c}, t)) \in \text{WN}^{-e}$ for all $\mathbf{c} \in \text{RED}(?C)^\perp$ and $t \in \text{RED}(A)^\perp$; and we want to conclude that $\text{sl}(\text{CUT}(\beta; \mathbf{c}, u)) \in \text{WN}^{-e}$ for all $\mathbf{c} \in \text{RED}(?C)^\perp$ and for all u in $\text{RED}(!A)^\perp$. Now, it is easy to show that we can make a simplification on the whole sequence \mathbf{c} , namely that \mathbf{c} is of the form $!d$, for $d \in \text{RED}(\mathbf{C})^\perp$. By several $(!/?p)$ reduction steps we reduce $\text{CUT}(\beta; \mathbf{c}, u)$ to $\text{CUT}(!\text{CUT}(\beta'; \mathbf{c}), u)$. Now, by induction hypothesis $\text{CUT}(\beta'; \mathbf{c}) \in \text{RED}(A)$, hence $!\text{CUT}(\beta'; \mathbf{c}) \in !\text{RED}(A) \subset \text{RED}(!A)$. Thus $\text{sl}(\text{CUT}(!\text{CUT}(\beta'; \mathbf{c}), u)) \in \text{WN}^{-e}$. Finally, since $\text{sl}(\text{CUT}(\beta; \mathbf{c}, u)) \rightsquigarrow_{(!/?p)^*}$

$\text{sl}(CUT(!CUT(\beta'; \mathbf{c}), u))$, we conclude that $\text{sl}(CUT(\beta; \mathbf{c}, u)) \in \text{WN}^{-e}$.

□

We can eventually conclude:

Theorem 4.10 (SN theorem) *If β is a net, then $\beta \in \text{SN}$.*

PROOF. If β is a net, then $\text{sl}(\beta)$ satisfies *AC* (Proposition 4.1) and $\text{sl}(\beta) \in \text{WN}^{-e}$ (Theorem 4.9). We can then apply Theorem 3.2 and prove that $\text{sl}(\beta) \in \text{SN}$, from which we conclude that $\beta \in \text{SN}$ (Proposition 4.2). □

Acknowledgements: We thank Jean-Yves Girard for his course on proof-theory given in Roma Tre in fall 2004, where he presented Gandy's method for natural deduction. We also thank Paolo Tranquilli, Olivier Laurent and Michele Abrusci for several discussions on the subject.

References

- [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.
- [DCKP03] Roberto Di Cosmo, Delia Kesner, and Emmanuel Polonovski. Proof nets and explicit substitutions. *Mathematical Structures in Computer Science*, 13(3):409–450, June 2003.
- [DJS97] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. A new deconstructive logic: linear logic. *JSL*, 62(3):755–807, September 1997.
- [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'état, 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.
- [Gir91] Jean-Yves Girard. Quantifiers in linear logic II. In Corsi and Sambin, editors, *Nuovi problemi della logica e della filosofia della scienza*, pages 79–90, Bologna, 1991. CLUEB.
- [Gir96] Jean-Yves Girard. Proof-nets: the parallel syntax for proof-theory. In Aldo Ursini and Paolo Agliano, editors, *Logic and Algebra*, volume 180 of *Lecture Notes In Pure and Applied Mathematics*, pages 97–124, New York, 1996. Marcel Dekker.
- [HvG03] Dominic Hughes and Rob van Glabbeek. Proof nets for unit-free multiplicative-additive linear logic. In *Proc. of the eighteenth annual symposium on Logic In Comp. Science*, pages 1–10, Ottawa, June 2003. IEEE, IEEE Comp. Soc. Press.
- [Klo92] Jan Willem Klop. Term rewriting systems. In Samson Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 1–116. IEEE, Oxford University Press, June 1992.
- [Kri90] Jean-Louis Krivine. *Lambda-Calcul : Types et Modèles*. Études et Recherches en Informatique. Masson, 1990.
- [Lau02] Olivier Laurent. *Étude de la polarisation en logique*. Thèse de doctorat, Université Aix-Marseille II, March 2002.
- [LQTdF05] Olivier Laurent, Myriam Quatrini, and Lorenzo Tortora de Falco. Polarized and focalized linear and classical proofs. *Annals of Pure and Applied Logic*, 134(2–3):217–264, July 2005.

- [LTdF04] Olivier Laurent and Lorenzo Tortora de Falco. Slicing polarized additive normalization. In Thomas Ehrhard, Jean-Yves Girard, Paul Ruet, and Philip Scott, editors, *Linear Logic in Computer Science*, volume 316 of *London Mathematical Society Lecture Note Series*, pages 247–282. Cambridge University Press, November 2004.
- [LTdF06] Olivier Laurent and Lorenzo Tortora de Falco. Obsessional cliques: a semantic characterization of bounded time complexity. In *Proceedings of the twenty-first annual IEEE symposium on Logic In Computer Science (LICS '06)*, 2006.
- [Oka99] Mitsuhiro Okada. Phase semantic cut-elimination and normalization proofs of first- and higher-order linear logic. *Theor. Comput. Sci.*, 227(1-2):333–396, 1999.
- [Pag08] Michele Pagani. Between interaction and semantics: visible acyclic nets. In preparation, 2008.
- [PS08] Michele Pagani and Alexis Saurin. Stream associative nets and lambda-calculus. Research Report 6431, INRIA, 02 2008. Submitted to *Theor. Comput. Sci.*, special issue for the 60th birthday of Jean-Yves Girard.
- [Py98] Walter Py. *Confluence en $\lambda\mu$ -calcul*. Thèse de doctorat, Université de Savoie, January 1998.
- [Reg92] Laurent Regnier. *Lambda-Calcul et Réseaux*. Thèse de doctorat, Université Paris VII, 1992.
- [Sø97] Morten Heine Sørensen. SN from WN in typed λ -calculi. *Information and Computation*, 133(1):35–71, 1997.
- [TdF00] Lorenzo Tortora de Falco. *Réseaux, cohérence et expériences obsessionnelles*. Thèse de doctorat, Université Paris VII, January 2000.
- [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.
- [Tra08] Paolo Tranquilli. Intuitionistic differential nets and lambda calculus. Submitted to: *Theoretical Computer Science*, 2008.