# EXPONENTIALS AS SUBSTITUTIONS,
# AND THE COST OF CUT ELIMINATION IN LINEAR LOGIC

BENIAMINO ACCATTOLI

Inria & LIX, École Polytechnique, France
*e-mail address*: beniamino.accattoli@inria.fr

ABSTRACT. This paper introduces the *exponential substitution calculus* (ESC), a new presentation of cut elimination for IMELL, based on proof terms and building on the idea that exponentials can be seen as explicit substitutions. The idea in itself is not new, but here it is pushed to a new level, inspired by Accattoli and Kesner's linear substitution calculus (LSC).

One of the key properties of the LSC is that it naturally models the sub-term property of abstract machines, which is the key ingredient for the study of reasonable time cost models for the $\lambda$-calculus. The new ESC is then used to design a cut elimination strategy with the sub-term property, providing the first polynomial cost model for cut elimination with unconstrained exponentials.

For the ESC, we also prove untyped confluence and typed strong normalization, showing that it is an alternative to proof nets for an advanced study of cut elimination.

## 1. INTRODUCTION

Two key aspects of linear logic are its resource-awareness and that it models the evaluation of $\lambda$-terms via cut elimination, even of untyped $\lambda$-terms, if recursive formulas are allowed. One would then expect that, given a $\lambda$-term $t$ represented as a linear proof $\pi_t$, the length of cut elimination in $\pi_t$ could provide estimates about the time complexity of $t$. Surprisingly, this is not (yet) the case.

**Linear Logic and Complexity Classes.** Linear logic is often used in *implicit computational complexity*, a field that aims at characterizing complexity classes with no explicit references to machine models. In this line of work, classes are characterized by seeing program execution as cut elimination in fragments of, and variations on, linear logic, usually obtained by constraining the exponential connectives in some way. Some representative papers are [GSS92, Gir98, DJ03, Laf04, MT03].

A first limitation of these results is that the bound is given on *whole* fragments of linear logic, without saying how to compute the cost of a fixed proof, which might be much lower than the bound for the fragment. A second limitation is that, as soon as one steps out of the fragment that characterizes the class, nothing is known.

**The Cut Elimination Clock.** The underlying problem is that there are no known complexity measures for cut elimination in linear logic. The use of linear logic for implicit complexity is so implicit that, not only it avoids machine models, it also never states what is the underlying logical unit for time, that is, what are the *ticks* of the cut elimination *clock*.

Formally, what is missing is a *polynomial time cost model* for cut elimination. That is, a cut elimination strategy of which *the number of steps* is a bound, up to a polynomial overhead, to the time complexity of implementing cut elimination according to that strategy. But be careful, it is not the number of steps that has to be polynomial: it would not be possible, linear logic cut elimination can take way more than polynomially many steps (in the size of the initial proof). It is the cost of implementing the sequence of steps (on a random access machine, or on another reasonable framework) that must be polynomial—ideally linear—in their number (which can be whatever) and in the size of the proof. In this way, the number of steps taken by the strategy on a proof $\pi$ provides a reliable measure for the time cost of $\pi$. Roughly, the steps of such a strategy become the ticks of a cut elimination clock.

One might wonder why not any strategy would do. The point is that cut elimination strategies often suffer of *size explosion*: one can build families of proofs $\{\pi_n\}_{n\in\mathbb{N}}$ on which the strategy iterates duplications in malicious ways, leading to a growth of the proof size which is exponential in the number $k$ of cut elimination steps. Therefore, $k$ cannot be taken as the time cost for implementing the cut elimination sequence (not even up to a polynomial), if the sequence has to be implemented as it is. Intuitively, taking such strategies as clocks would correspond to having irregular ticks of non-uniform length, some of which take a very long time. In Sect. 3, we shall show that cut elimination *by levels*, the strategy of reference in linear logic (especially for implicit computational complexity), suffers from size explosion.

The aim of this paper is to provide a clock of which the ticks are regular enough to serve as a reliable time measure for unconstrained exponentials. That is, we shall provide a new strategy of which the number of steps is a polynomial time cost model. In our pursuit, we are inspired by developments in the study of cost models for the $\lambda$-calculus and in the modern theory of $\lambda$-calculi with explicit substitutions.

**The Linear Substitution Calculus.** In the last decade, the study of *reasonable* time cost models, that is, of polynomial cost models that *additionally* are equivalent to the time cost model of Turing machines, has advanced considerably in the sister field of $\lambda$-calculus, starting with Accattoli and Dal Lago's result about the number of steps of the leftmost strategy [ADL16]. The advances have been enabled and developed over Accattoli and Kesner's *linear substitution calculus* (shortened to LSC) [Mil06, Acc12, ABKL14], which is a neat and compact refinement of the $\lambda$-calculus that probably can be considered the answer to the quest for the canonical $\lambda$-calculus with explicit substitutions (shortened to ES). The LSC refines ES with ideas from both proof nets, namely using *contextual* rewriting rules on terms—also called *at a distance*—to avoid commuting constructors (taken from Accattoli and Kesner [AK10]), and the $\pi$-calculus/bigraphs, in modeling duplication as replication in $\pi$ (following Milner [Mil06]).

**The Sub-Term Property.** The relevance of the LSC for cost models is due to its natural modeling of strategies having the *sub-term property* often found in abstract machines:

*All sub-terms duplicated or erased along an evaluation sequence from $t$ are sub-terms of $t$.*

When a strategy $\to_{\mathtt{st}}$ has the sub-term property, the cost of implementing a sequence $t \to_{\mathtt{st}}^{k} s$ is polynomial, and in general *linear*, in $k$ and in the size $|t|$ of $t$. Therefore, the sub-term property implies that the number of steps is a polynomial cost model for time. Intuitively, it states that the steps of the strategy are the *regular ticks* of a reliable clock. It represents, for the study of cost models, what the sub-formula property is for proof-search, or the cut-elimination theorem for sequent calculi.

Typically, important strategies of the $\lambda$-calculus such as weak head, head, or leftmost-outermost reduction do not have the sub-term property (actually, we show in Sect. 3 that no strategy in the $\lambda$-calculus has the property), while their analogous in the LSC does. One of the key points is that the sub-term property requires *micro-steps* evaluation, that is, performing one variable replacement at a time (as in the LSC), rather than small-step evaluation, that is, using rules resting on meta-level substitution such as $\beta$. The sub-term property and the related degeneracy of size explosion are discussed in detail in Sect. 3.

**Linear Logic and the Sub-Term Property.** In linear logic, the problem is not just that there are no known cut elimination strategies providing polynomial cost models. What is worse, is that there are no known micro-step strategies with the sub-term property (as we discuss below), despite the clear linear logical flavor of the property. Therefore, the study of cost models is, at present, simply *out of reach*.

How can one recover the sub-term property in linear logic? This is the challenge addressed by this paper. There are two natural possible routes. One is working inside linear logic as it is usually presented, and try to recover the sub-term property. This would inevitably mean working with proof nets, as cut elimination in the sequent calculus requires to deal with too many commutative cases. Another one is to develop an alternative, *commutation-free* presentation of linear logic akin to the LSC, and use it to design a strategy with the sub-term property. We choose the second option, for three reasons.

(1) *Terms are more easily manageable*: graphical syntaxes are too hard to manage for the long and delicate proofs typical of the study of cost models.
(2) *Better rewriting*: the rewriting theory of the LSC is better behaved than the one of proof nets.
(3) *Novelty*: we aim at a fresh look at linear logic, importing ideas from a sister field.

**This Paper.** We provide three main contributions:

(1) *Design*: we introduce the *exponential substitution calculus* (ESC), which is a new presentation of linear logic cut elimination based over the theory of ES and the LSC and free from commutative cases. It is a design study, done in a principled way. It is also a *stress test*, as linear logic has many more constructors and rewriting rules than the LSC. We shall see that, in order to guarantee some expected properties of ES, one is forced to turn to *intuitionistic* linear logic.
(2) *Sub-term strategy*: we define a new cut elimination strategy for the ESC and prove that it has the sub-term property. Therefore, we solve the issue mentioned at the beginning of the paper, providing the first polynomial time cost model for unconstrained exponentials.
(3) *Foundations*: we prove various key properties of the ESC, among which untyped confluence and typed strong normalization, thus providing solid foundations for our presentation. The non-trivial proofs of these properties are developed from scratch, using elegant proof techniques developed for the LSC.

The next section provides overviews of these contributions.

**Related Work.** To our knowledge, there are no works in the literature studying cost models for linear logic. Many term and process calculi for or related to linear logic have been proposed, for instance [LM92, Abr93, BBdPH93, Wad93, BS94, BW96, RR97, MOTW99, Sim05, OH06, CP10, Wad12, CFM16, EG16]. There are also proposals for $\lambda$-calculi with ES with linear features such as [GdPR00, KL07, FS14]. None of these calculi employs rewriting rules at a distance as we do here. Mazza uses a natural deduction linear calculus at a distance [Maz17], and Kesner develops a $\lambda$-calculus with ES reflecting proof nets cut elimination [Kes22] but only for the fragment representing the $\lambda$-calculus. Various authors consider terms for proof *structures* [Abr93, FM99, MS08, Ehr14, CA21] which then need correctness criteria, not required here.

The sub-term property is a folklore property first called as such by Accattoli and Dal Lago [ADL12], who also show a surprising link with the standardization theorem in [ADL16].

**Journal Version and Proofs.** This journal version extends the conference one adding explanations, in particular Sect. 3 is new, and most proofs together with the intermediary lemmas that were omitted from the conference version. A few proofs are particularly long and tedious so they are still omitted or partially omitted, but can be found on Arvix in the technical report [Acc22] associated to the conference paper, which is accessible as version one ("v1", see the bibliography entry for the link) of the present paper on Arxiv (which is instead "v2").

## 2. OVERVIEW OF THE CONTRIBUTIONS

2.1. **Contribution 1: The Exponential Substitution Calculus.** By allowing duplication and erasure of sub-proofs, the exponentials can be seen as a substitution device: it is the core of the simulation of the $\lambda$-calculus into linear logic due to Girard [Gir87]. Historically, ES became popular with a calculus by Abadi et al. [ACCL91], later shown defective at the rewriting level by Melliès [Mel95]. Linear logic and the exponentials were thus considered as a more solid formalism to borrow from. Di Cosmo and Kesner were the first ones to do so [CK97], seeing ES as exponential cuts.

Along the years, however, the theory of ES made progresses of its own, and here we reverse the transfer. The idea is enforcing and pushing to the extreme the slogan:

$$\text{Exponentials} = \text{Explicit Substitutions}$$

To continue, we need to recall the very basics of ES.

**Basics of Explicit Substitutions.** The idea is to extend the $\lambda$-calculus with a new term constructor $t[x \leftarrow s]$ (which is just a compact notation for let $x = s$ in $t$, but with no fixed order of evaluation between $s$ and $t$) denoting a delayed or *explicit* substitution, and decomposing the $\beta$-rule:

$$(\lambda x.t)s \rightarrow_\beta t\{x \leftarrow s\}$$

where $t\{x \leftarrow s\}$ is meta-level substitution, into two rules:

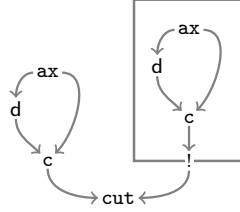$$(\lambda x.t)s \rightarrow_{\mathtt{B}} t[x \leftarrow s] \rightarrow_{\mathtt{s}} t\{x \leftarrow s\}$$

Figure 1: Untyped proof net of classical LL reducing to itself in 3 micro steps.

Now, rule $\to_\mathtt{s}$ is further decomposed into various *micro-step* rules. There are many possible sets of such substitution rules. What they all have in common, even the defective ones mentioned above, is that, when considered *separately* from rule $\to_\mathtt{B}$, they are strongly normalizing even without types.

**Design Principles.** Our *exponentials as substitutions* design of the ESC is based on the following principles.

*Principle 1: proofs as typing derivations.* Calculi with ES are studied as *untyped* calculi. Then we want the ESC to be an untyped calculus with good properties by itself, and see proofs of linear logic as typing derivations for it.

*Principle 2: ES terminates/intuitionism.* As we recalled above, the rules that manage ES are always strongly normalizing without types. It is possible to define also *untyped* linear logic proofs, as done e.g. in de Carvalho et al. [dCPTdF11], obtaining a notion of untyped proof net. In the case of classical linear logic *à la* Girard (with involutive negation), the exponentials of such proof nets can diverge by themselves, as in Fig. 1 (left). In such a setting, then, exponentials are *not* as ES. We then switch to *intuitionistic* linear logic, and prove that therein untyped exponentials are strongly normalizing (Theorem 8.8, p. 32). To our knowledge, this is a new result and the first time that such a discrepancy between the classical and intuitionistic case is pointed out. The proof of the theorem requires long calculuations but it is otherwise simple. Termination is obtained via a measure defined by induction over terms. Essentially, the result states that in the intuitionistic setting *exponentials are as ES*. It is one of the main contributions of the paper.

*Principle 3: micro and small steps together.* The sub-term property requires a micro-step operational semantics. But studying micro-step ES constantly requires to refer to meta-level substitution, which is small-step. For linear logic, Girard originally considered micro-step cut elimination [Gir87], but soon afterwards Regnier introduced a small-step variant, to study the $\lambda$-calculus [Reg92], and both are frequently used in the literature. In the ESC, we want small and micro steps to *co-exist*, so as to provide a comprehensive framework.

**Left Splitting.** The design of the ESC systematically exploits a simple fact about IMELL proofs: they can be seen as ending on a sequence of left rules following a right rule or an axiom. Representing a proof as a term $t$, one can thus uniquely split it as $t = L\langle v \rangle$, that is, a sub-proof ending on a right rule or an axiom, what we shall call a *value $v$*, and a possibly empty sequence of left rules $L$ from the last sequent of $v$ to the last sequent of $t$. Such a *left splitting* is a basic property of sequent proofs that is not related to polarity or focussing—it is in fact simpler—and yet plays a crucial role in our rewriting rules at a distance for IMELL.

**Additives.** We do not consider the additive connectives, and so we deal with IMELL. The reason is that they are both trivial and challenging, depending on the approach. If one adopts *slices* [Gir87, LTdF04], then our results lift smoothly. Without slices, instead, it is unclear how to combine additives and LSC-style micro steps while retaining all the properties.

**Syntactical Variants.** A number of alternative syntaxes could have been adopted, typically Curien & Herbelin $\overline{\lambda}\mu\tilde{\mu}$ calculi [CH00, CFM16], Benton's linear-non-linear approach [Ben94], Pfenning & Caires' processes [CP10], and, of course, proof nets or natural deduction. We prefer avoiding them to show that their additional ingredients (the distinguished conclusion on the left of sequents [CH00, CFM16], the linear-non-linear separation [Ben94], and the graphical syntax) are not required for studying cost models for linear logic, while we avoided processes because their evaluation does not compute cut-free proofs, since they do not evaluate under prefixes. Finally, we prefer the sequent calculus to natural deduction because it is more commonly used for presenting linear logic, and because in this way we provide a novel use of the LSC technology, given that the LSC is based on natural deduction. Our study can however be adapted to any of these settings.

2.2. **Contribution 2: the Sub-Term Strategy.** The proof nets strategy of reference is the notion of *least level* (or *by levels*) strategy $\to_{ll}$ (introduced—we believe—by Girard [Gir98] and studied for instance by de Carvalho et al. [dCPTdF11]), which reduces cuts of minimal level, defined as the number of !-boxes surrounding a cut. Unfortunately, $\to_{ll}$ is not a good candidate for a linear logic clock. Consider the following diagram made out of micro steps at level 0, which is an important sub-strategy $\to_{l_0}$ of $\to_{ll}$:
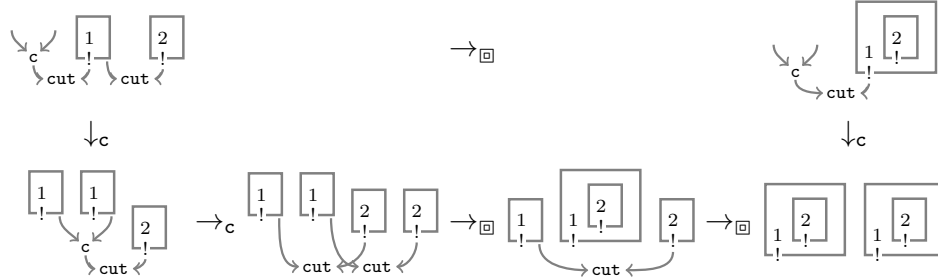


The right-down path shows that $\to_{l_0}$, and thus $\to_{ll}$, lacks the sub-term property, as the duplicated box is not a sub-proof of the initial one. Moreover, iterating such a pattern gives size explosion for $\to_{l_0}$, as shall we show in Sect. 3. The down-left path has the sub-term property but it shows that different $\to_{l_0}$ paths can have different lengths—that is, $\to_{l_0}$ is not *diamond*—forbidding to take its number of steps as a measure, because the number of its steps to normal form is an ambiguously defined quantity.

**The Linear Head Strategy.** There actually is a micro-step strategy in the linear logic literature which is both diamond and with the sub-term property. It is Mascari & Pedicini's and Danos & Regnier's *linear head strategy* $\to_{\text{lh}}$ [MP94, DR04] (which is a sub-strategy of level 0 cut elimination) but it is defined only on the fragment representing the $\lambda$-calculus and it does not compute cut-free proofs.

The linear head strategy is naturally modeled by the LSC, used by Accattoli and co-authors for proving properties of $\to_{\text{lh}}$, design variants and extensions [Acc12, ABKL14, ABM14, ABM15], as well as to prove that it provides a reasonable time cost model [ADL12].

**The Good Strategy.** The ESC strategy introduced here is simply called *the good strategy* $\to_{\mathsf{G}}$, because it avoids *bad* steps breaking the sub-term property. It is the generalization of the linear head strategy to the whole of IMELL and extended as to compute cut-free proofs. The good strategy is micro-step, diamond, and it has the sub-term property, and thus provides the first polynomial cost model for IMELL[1]. Its design is based on the *creation of cuts*, a notion due to the intuitionistic arrow $\multimap$ and invisible in classical linear logic.

2.3. **Overview of Contribution 3: Foundations.** We provide a foundational study of the ESC, ensuring that the new cut elimination rules are well behaved, but also to promote it as an alternative to the use of proof nets for IMELL. We prove three important properties:

(1) *Untyped confluence*: the *untyped* case is possibly divergent and thus more difficult than the typed case, as one cannot exploit termination in the proof technique. We prove confluence using the Hindley-Rosen method, and exploit termination nonetheless by building on the strong normalization of the exponentials, following an approach pushed forward by Accattoli in the study of the LSC [Acc12].

(2) *Untyped PSN*: preservation of untyped strong normalization (shortened to PSN) is a property typical of calculi with ES, stating that if a term is SN for the small-step rules (in the untyped setting where terms can also be divergent) then it is SN for the micro-step rules. Essentially, it states that the decomposition of small steps in micro steps does not introduce degeneracies. The proof is based on a technique due to Kesner [Kes09]. This theorem is notoriously technical for $\lambda$-calculi with ES. We adapt a simple proof by Accattoli and Kesner [AK10], where the simplicity is enabled by adopting rules at a distance. Our adaptation is even simpler than the proof in [AK10].

(3) *Typed SN*: we prove strong normalization (SN) of the small-step rules in the typed case, using the reducibility method. This theorem too is notoriously technical to prove for fragments of linear logic including the exponentials. We here provide what is probably the simplest and cleanest proof of SN in the literature, improving the already simplified approach of Accattoli [Acc13] based on rules at a distance. We then use PSN to transport SN to micro steps.

Confluence and strong normalization are not usually studied for sequent calculi with traditional cut elimination, that is, with both principal and commutative cases, because of the following two facts:

- *No SN*: cut elimination is *not* SN, because cut commutes with itself, leading to non-termination:

$$\cfrac{\Gamma \vdash B \qquad \cfrac{\Pi \vdash A \qquad \Delta, A, B \vdash C}{\Delta, \Pi, B \vdash C}\ cut}{\Delta, \Pi, \Gamma, \vdash C}\ cut \quad \to \quad \cfrac{\Pi \vdash A \qquad \cfrac{\Gamma \vdash B \qquad \Delta, A, B \vdash C}{\Delta, A, \Gamma \vdash C}\ cut}{\Delta, \Pi, \Gamma \vdash C}\ cut$$

- *No confluence*: cut elimination is not confluent, not even in IMLL (!), because commutations affect the result, as shown by this counter-example, courtesy of Olivier Laurent:

---

[1] Is it a *reasonable* cost model? Roughly, yes. For time reasonability, the subtle part is polynomiality of the cost model, that is, the polynomial simulation of the ESC on Turing machines/RAM. The complementary part—here missing—is simulating Turing machines in the untyped ESC within polynomial overhead. It is true, but for very minor reasons it does not follow from results in the literature and it has to be (tediously) reproved from scratch. Moreover, the missing part is not relevant/cannot hold for the *typed* ESC, as IMELL is not Turing-complete (IMELL cut-elimination being strongly normalizing, it cannot model diverging computations).

$$\cfrac{\cfrac{\overline{A \vdash A} \quad \overline{B \vdash B}}{A, A \multimap B \vdash B} \multimap_l \quad \cfrac{\overline{B \vdash B} \quad \overline{C \vdash C}}{B, B \multimap C \vdash C} \multimap_l}{A, A \multimap B, B \multimap C \vdash C} \text{cut}$$

$$+ \swarrow \qquad \searrow +$$

$$\cfrac{\overline{A \vdash A} \quad \cfrac{\overline{B \vdash B} \quad \overline{C \vdash C}}{B, B \multimap C \vdash C} \multimap_l}{A, A \multimap B, B \multimap C \vdash C} \multimap_l \qquad \cfrac{\cfrac{\overline{A \vdash A} \quad \overline{B \vdash B}}{A, A \multimap B \vdash B} \multimap_l \quad \overline{C \vdash C}}{A, A \multimap B, B \multimap C \vdash C} \multimap_l$$

Retrieving confluence and SN requires cut elimination *modulo commutations*, or via proof nets, or adding some rigidity to proofs such as focalization. In stark contrast, the cut elimination *at a distance* of the ESC, being free from commutations, achieves them without rewriting modulo nor modifying the deductive system.

Our study thus lies a new foundation for the study of cut elimination. It also sums up ten years of research on the rewriting of the LSC, at the same time generalizing the developed techniques to the considerably more general setting of IMELL.

## 3. Sub-Term Property and Size Explosion

The sub-term property is used in many works on abstract machines, or the LSC, or reasonable cost models for the $\lambda$-calculus. Here we aim at providing evidence of its relevance. We first discuss it in the $\lambda$-calculus and then see an example of size explosion in proof nets.

We give a new and more accurate definition of the property than the one in the introduction. We actually define two properties, a structural one and a quantitative one.

**Definition 3.1** (Sub-term property)**.** A rewriting system $(S, \rightarrow)$ has the sub-term property if, for every $t_0 \in S$ and every sequence $t_0 \rightarrow^n t_n$, every step of the sequence either only involves a constant number of constructors or it duplicates or erases a term:

- *Structural sub-term property*: which is a sub-term of $t_0$ up to variables renaming;
- *Quantitative sub-term property*: of size bound by $|t_0|$.

In the case of duplications, the number of copies is also bound by $|t_0|$ (for both notions).

The structural property provides the intuition and gives the name to the property. For cost analyses, however, one only needs the quantitative version. The two formulations are two sides of the same concept and most of the time we simply refer to the sub-term property, without further specification.

**$\lambda$-Calculus *vs* the Sub-Term Property.** In the $\lambda$-calculus, no evaluation strategy has the structural sub-term property—as we now show—because its operational semantics is somewhat too coarse. Let $\tau_t := \lambda y.ytt$ and $\mathtt{I} := \lambda x.x$. Then:

$$s := \begin{array}{ll} (\lambda x.x(\lambda z.\tau_z)\tau_x)\mathtt{I} & \rightarrow_\beta \quad \mathtt{I}(\lambda z.\tau_z)\tau_{\mathtt{I}} \quad \rightarrow_\beta \\ (\lambda z.\tau_z)\tau_{\mathtt{I}} & \rightarrow_\beta \quad \tau_{\tau_{\mathtt{I}}} \end{array} \tag{3.1}$$

Note that the term $\tau_{\mathtt{I}}$ duplicated by the third step is not a sub-term of $s$. Moreover, $s$ is closed and each term in the sequence as at most one $\beta$-redex, which is out of abstractions and the argument of which is a value. Therefore, the example affects also closed weak evaluation, and even its call-by-value variant.

What actually happens is that in the $\lambda$-calculus duplicated sub-terms are *combinations* of sub-terms of the initial term, as it is the case for $\tau_{\mathtt{I}}$ in the example above. The size of such

combinations can grow exponentially with the number of $\beta$-steps, a degeneracy called *size explosion* and affecting *all* strategies in the $\lambda$-calculus, as we shall now recall, suggesting that the number of $\beta$-steps might *not* be a polynomial cost model for *any* strategy.

**Inevitable Size Exploding Family.** In the $\lambda$-calculus, there is a size exploding family that explodes no matter the evaluation strategy. Its definition is in two steps. Firstly, define the following *pre-family* $\{t_n\}_{1 \leq n \in \mathbb{N}}$, where, again, $\tau_t := \lambda y.ytt$:

$$t_1 \quad := \quad \lambda x.\tau_x \qquad\qquad t_{n+1} \quad := \quad \lambda x.t_n\tau_x$$

The size exploding family is actually given by $\{t_n\mathtt{I}\}_{1 \leq n \in \mathbb{N}}$ , that is, it is obtained by applying the term $t_n$ of the pre-family to the identity. Note that it is a family of closed terms. We also define the family of results $\{r_n\}_{n \in \mathbb{N}}$ as follows:

$$r_0 := \mathtt{I} \qquad\qquad r_{n+1} := \tau_{r_n}$$

Last, let $\rightarrow_{wh}$ be the weak head strategy, which reduces only head redexes out of abstractions and is the simplest meaningful reduction strategy in the $\lambda$-calculus.

**Proposition 3.2** (Closed and strategy-independent size explosion)**.** *Let $n > 0$. Then*

(1) *Size explosion for the weak head strategy*: $t_n\mathtt{I} \rightarrow^n_{wh} r_n$ *and then $i$-th step of the sequence makes two copies of $r_{i-1}$.*

(2) *Strategy independent: any other reduction sequence $t_n\mathtt{I} \rightarrow^*_{wh} r_n$ has length $n$, and all redexes in these sequences are $\beta_v$ redexes.*

*Moreover, $|t_n\mathtt{I}| = \mathcal{O}(n)$, $|r_n| = \Omega(2^n)$, $t_n\mathtt{I}$ is closed, and $r_n$ is a $\beta$ normal form.*

*Proof.* The bounds on the sizes are straightforward inductions.

(1) We prove a more general statement: $t_n r_m \rightarrow^n_{wh} r_{n+m}$ (note that $r_1 = \mathtt{I}$). By induction on $n$. If $n = 1$ then $(\lambda x.\tau_x)r_m \rightarrow_{wh} \tau_{r_m} = r_{m+1}$. For $n+1$, $t_{n+1}r_m = (\lambda x.t_n\tau_x)r_m \rightarrow_{wh} t_n\tau_{r_m} = t_n r_{m+1}$. By *i.h.*, $t_n r_{m+1} \rightarrow^n_{wh} r_{n+m+1}$.

(2) It is easily seen that the $n$ redexes in $t_n\mathtt{I}$ are all hereditarily independent and by-value: they cannot duplicate/erase each other, their arguments are abstractions, and they all produce terms with the same property. Then all local confluence diagrams from $t_n\mathtt{I}$, as well as from the terms reachable from it, are diamonds and all reductions to normal form have the same length. □
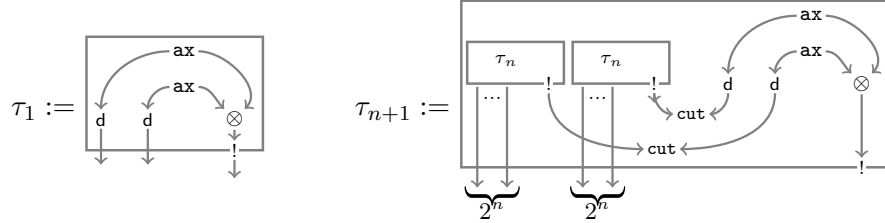
Size explosion is the fact that the linear size term $t_n\mathtt{I}$ reduces in $n$ steps to the exponential size result $r_n$, thus doing an exponential amount of work in a linear number of steps. The first point specifies that $t_n\mathtt{I}$ explodes while using the weak head strategy, *and* that in this case each step of the sequence (but the first one) breaks the structural sub-term property, as it duplicates $r_n$ which is *not* a sub-term of the initial term $t_n\mathtt{I}$. Thus, breaking the sub-term property leads to size explosion for the weak head strategy, which breaks the *quantitative* sub-term property, as, for $n$ big enough, the last duplication of the sequence involves a term bigger than the initial one.

The second point says that the family evaluates to the same result and in the same number of steps according to any other evaluation strategy, no matter whether strong or weak, call-by-name or call-by-value. It is easily seen that the family is also typable with simple types, although of types that grow in size exponentially in $n$. Since the family is also closed, size explosion affects all the main variants and dialects of the $\lambda$-calculus.

Let us show a final observation about size explosion. Consider the third element $t_3\mathtt{I}$ of the given family, and let us reduce it in a *innermost way*:

$$t_3\mathtt{I} \;=\; (\lambda x.(\lambda x.(\lambda x.\tau_x)\tau_x)\tau_x)\mathtt{I} \;\to_\beta\; (\lambda x.(\lambda x.\tau_{\tau_x}))\tau_x)\mathtt{I} \;\to_\beta\; (\lambda x.\tau_{\tau_{\tau_x}})\mathtt{I} \;\to_\beta\; \tau_{\tau_{\tau_\mathtt{I}}} \qquad (3.2)$$

The first two steps duplicate $\tau_x$ and that the last step duplicates $\mathtt{I}$: it looks like the structural sub-term property is not broken. Here it plays a role the final clause in the definition of the property: *in the case of duplications, the number of copies is also bound by* $|t_0|$. Indeed in (3.2) it is the number of copies of the duplicated sub-term that grows exponentially (note that $\tau_{\tau_{\tau_{x_3}}}$ has eight occurrences of $x_3$), thus still breaking the sub-term property.

**Small-Steps** *vs* **Micro-Steps.** The reader might be aware of the fact that, despite what just explained, the number of steps of some strategies of the $\lambda$-calculus do provide polynomial cost models. How is that possible?

There is a way out, but it requires deviating from the $\lambda$-calculus. The lack of the sub-term property in the $\lambda$-calculus stems from its *small-step* operational semantics, $\beta$-reduction, which is based on meta-level substitution. To retrieve the property, one has to add ES (or some other form of sharing) to the $\lambda$-calculus and decompose $\beta$-reduction in *micro-steps* performing *one* variable replacement at a time. Then, there are strategies for the refined setting that have the sub-term property, of which the number of steps thus is a polynomial cost model. The micro-step refinement can be formulated in various ways, the LSC being an elegant one. Without going into too much detail, the result is then transferred to the $\lambda$-calculus, by showing that there is a polynomial simulation (up to some form of sharing) of a small-step strategy by a micro-step one with the sub-term property. Therefore, small-step polynomiality is inherited from the micro-step formalism (as in Blelloch and Greiner [BG95], Sands et al. [SGM02], Dal Lago and Martini [DLM12], Accattoli and Dal Lago [ADL12, ADL16], Accattoli et al. [ACSC21], Biernacka et al. [BCD21]).

Summing up, the sub-term property seems to be *mandatory* for the study of cost models, because even when the cost model strategy does not have the property, the proof of its polynomiality rests on an auxiliary strategy that does indeed have it. Moreover, the micro-step aspect seems *unavoidable* for unlocking polynomial time cost models.

**Size Explosion in Linear Logic.** We now explain how to obtain a size exploding family for linear logic proof nets, taking as notion of evaluation *cut elimination at level 0* (the level of a link is the number of ! boxes in which it is contained), noted $\to_{l_0}$, which is, roughly, the analogous of head reduction for proof nets (all head reduction redexes are redexes at level 0, when $\lambda$-terms are translated to proof nets). We shall need only the following two reduction rules for proof nets:



We shall use $\to_{\mathsf{c}0}$ and $\to_{\boxdot 0}$ for when these rules are applied to cuts at level 0.

**Size Exploding Family.** For the time being, let's consider untyped proof nets. We now define various families of proof nets. The $n$-th net $\tau_n$ has the following shape:



And it is defined as follows:



Then we define the family $\pi_n$, which is obtained from $\tau_n$ by contracting all its $2^n$ auxiliary conclusions via a tree of contractions:
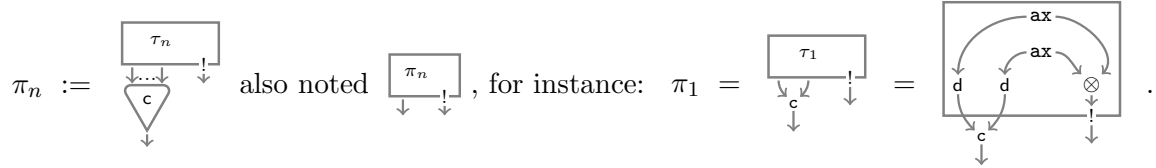


Finally, define the following family of nets:

$$\rho_1 \;:=\; \pi_1 \qquad\qquad \rho_{n+1} \;:=\; \text{}$$
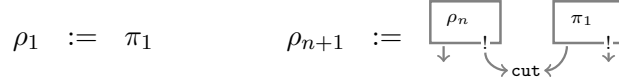
**Proposition 3.3** (Size explosion in MELL). $\rho_n \to_0^{3(n-1)} \pi_n$, $|\rho_n| = \mathcal{O}(n)$, and $|\pi_n| = \Omega(2^n)$.

*Proof.* By induction on $n$. The bounds on the sizes follow immediately from the definition and the *i.h.* For $n = 1$, we have $\rho_1 = \pi_1$, so the statement holds. For $n + 1$, we have that by *i.h.* $\rho_n \to_0^{3(n-1)} \pi_n$. Therefore, we obtain:
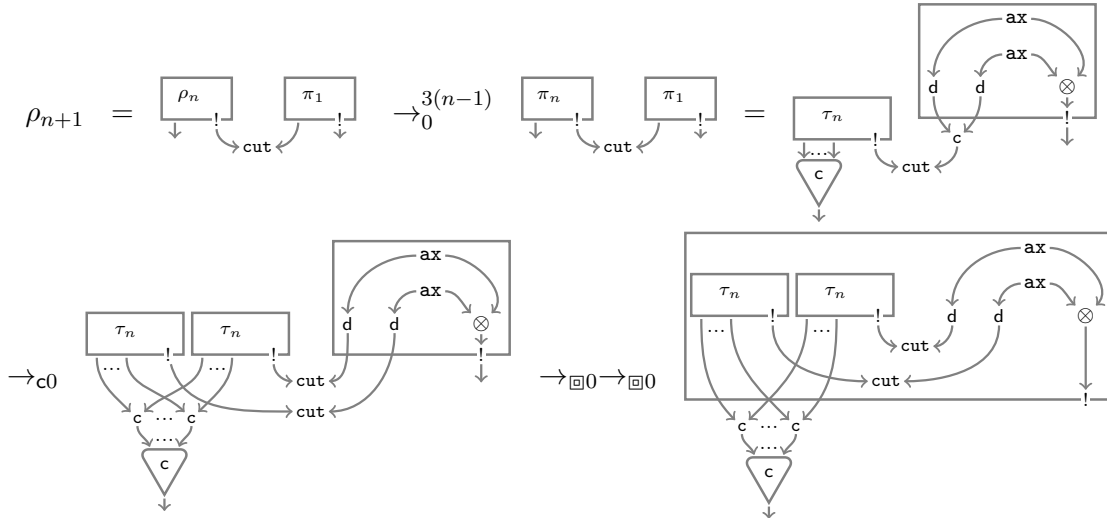
The size exploding family of Prop. 3.3 is a chain of what Roversi and Vercelli in [RV09] call *spindles*, which they identify as the problematic configurations for representing polynomial time.

**Typing the Size Exploding Family.** The proof net $\rho_1 = \pi_1$ used to build the family can be typed by assigning type $A$, for an arbitrary type $A$, to the right conclusion of the axioms (thus having $A^\perp$ on the left conclusion of the axioms), and $A_1 :=!(A \otimes A)$ for the right conclusions of $\rho_1$ (and $?A^\perp$ on the left conclusion of $\rho_1$). To type the second term of the family $\rho_2$, and assuming that the left occurrence of $\pi_1$ in $\rho_2$ is typed as just described, the right one needs to be typed using $A_1$ as type of the axioms, thus obtaining as type of the right conclusion, $A_2 :=!(A_1 \otimes A_1) =!(!(A \otimes A) \otimes !(A \otimes A))$. It is then easily seen that the type $A_{n+1}$ of the right conclusion of $\rho_{n+1}$ is given by $A_{n+1} =!(A_n \otimes A_n)$, that is, the type size grows exponentially with $n$.

**Size Explosion and Elementary Linear Logic.** The proofs $\rho_n$ of the size exploding family actually belong to *elementary linear logic* (ELL): they use derelictions—which are forbidden in ELL—but those derelictions can be seen as the auxiliary ports of functorial promotions associated to the !s, which are allowed in ELL. Therefore, size explosion affects ELL as well. We mention this fact because ELL has some interesting properties with respect to Lévy's *optimal reduction*. On the one hand, ELL provides a much simpler implementation of optimal reduction, because the technical *oracle* part of the implementation is not needed in ELL, as pointed out by Asperti [Asp98]. On the other hand, the complex behavior of duplication responsible for the *unreasonability* of optimal reduction is already fully present in ELL, and it is thus independent of the oracle, as shown by Asperti et al. [ACM00]. For the study of time cost models, restricting to E(ME)LL does not seem to simplify the problem, as size explosion affects both MELL and EMELL with respect to reduction at level 0. This is rather in line with the results of [ACM00].

## 4. The Linear Substitution Calculus

Here we recall the basics of the linear substitution calculus, which is the footprint of the exponential substitution calculus.

There are various ingredients. First, the syntax of the $\lambda$-calculus is extended with explicit substitutions $t[x \leftarrow s]$ (shortened to ES), which is a constructor binding $x$ in $t$ (meant to be substituted by $s$)—and a compact notation for let $x = s$ in $t$—while we use $t\{x \leftarrow s\}$ for meta-level substitution.

LSC TERMS  $t, s, u ::= x \mid \lambda x.t \mid ts \mid t[x \leftarrow s]$

The LSC is based on the rejection of commuting rules for ES. In traditional $\lambda$-calculi with ES, a term such as $(ts)[x \leftarrow u]$ would rewrite to $t[x \leftarrow u]s[x \leftarrow u]$ by commuting the ES and the application. In the LSC it does not: ES interact directly with variable occurrences, using *contexts* to specify a commutation-free rewriting system. Contexts, used pervasively in the study of the LSC, are terms with a *hole* $\langle \cdot \rangle$ intuitively standing for a removed sub-term.

Here, we need (general) contexts $C$ and the special case of substitution contexts $L$ (standing for *L*ist of substitutions).

$$\begin{array}{llll} \text{CONTEXTS} & C & ::= & \langle\cdot\rangle \mid Ct \mid tC \mid \lambda x.C \mid C[x\leftarrow t] \mid t[x\leftarrow C] \\ \text{SUB. CTXS} & L & ::= & \langle\cdot\rangle \mid L[x\leftarrow t] \end{array}$$

Replacing the hole of a context $C$ with a term $t$ (or another context $C'$) is called *plugging* and noted $C\langle t\rangle$ (resp. $C\langle C'\rangle$). In general, plugging does capture free variables, that is, $(\lambda x.\langle\cdot\rangle x)\langle x\rangle = \lambda x.xx$. We use $C\langle\!\langle t\rangle\!\rangle$ for a plugging with on-the-fly renaming of bound variables of $C$ to avoid capture in $t$. For instance, $(\lambda x.\langle\cdot\rangle x)\langle\!\langle x\rangle\!\rangle = \lambda y.xy$.

The LSC has three rewriting rules. As usual, the root rules are extended to be applied everywhere in a term via a context closure. An unusual aspect is that two of the rules also use contexts in the definition of the root rules.

<div align="center">

LSC ROOT REWRITING RULES

$$\begin{array}{lll} \text{BETA AT A DISTANCE} & L\langle\lambda x.t\rangle s & \mapsto_{\mathtt{dB}} \quad L\langle t[x\leftarrow s]\rangle \\ \text{LINEAR SUBSTITUTION} & C\langle\!\langle x\rangle\!\rangle[x\leftarrow s] & \mapsto_{\mathtt{ls}} \quad C\langle\!\langle s\rangle\!\rangle[x\leftarrow s] \\ \text{GARBAGE COLLECTION} & t[x\leftarrow s] & \mapsto_{\mathtt{gc}} \quad t \quad \text{if } x \notin \mathtt{fv}(t) \end{array}$$

$$\begin{array}{ll} \text{CONTEXTUAL CLOSURE:} & \dfrac{t \mapsto_a t'}{C\langle t\rangle \to_a C\langle t'\rangle} \\ (a \in \{\mathtt{dB}, \mathtt{ls}, \mathtt{gc}\}) \end{array}$$

</div>

For $\to_{\mathtt{ls}}$, the notation means that given $t[x\leftarrow s]$ with $x \in \mathtt{fv}(t)$ we consider a decomposition $t = C\langle\!\langle x\rangle\!\rangle$ isolating a free occurrence of $x$ (one among possibly many). The rule is inspired by replication in the $\pi$-calculus. Examples of steps:

$$\begin{array}{lll} (\lambda x.xy)[y\leftarrow t]su & \to_{\mathtt{dB}} & (xy)[x\leftarrow s][y\leftarrow t]u \\ \lambda x.((\lambda y.xx)[z\leftarrow t]) & \to_{\mathtt{gc}} & \lambda x.\lambda y.xx \\ (\lambda y.xx)[x\leftarrow t][z\leftarrow s] & \to_{\mathtt{ls}} & (\lambda y.xt)[x\leftarrow t][z\leftarrow s] \\ (\lambda y.xx)[x\leftarrow t][z\leftarrow s] & \to_{\mathtt{ls}} & (\lambda y.tx)[x\leftarrow t][z\leftarrow s] \end{array}$$

The LSC is a very well behaved rewriting system (see [Acc12, ABKL14, BB17]), conservatively extending the $\lambda$-calculus, and improving it in various respects.

**Micro-Step Simulates Small-Step.** Let $|t|_x$ the number of free occurrences of $x$ in $t$. The LSC simulates meta-level substitution with

$$t[x\leftarrow s] \to_{\mathtt{ls}}^{|t|_x} t\{x\leftarrow s\}[x\leftarrow s] \to_{\mathtt{gc}} t\{x\leftarrow s\}$$

that is, by first replacing one by one the occurrences of $x$ using $\to_{\mathtt{ls}}$, and then collecting $[x\leftarrow s]$ when none are left. Therefore, the LSC simulates $\beta$ reduction, as $(\lambda x.t)s \to_{\mathtt{dB}} t[x\leftarrow s] \to_{\mathtt{ls}}^* \to_{\mathtt{gc}} t\{x\leftarrow s\}$.

**Strong Normalization.** We define strong normalization as it is relevant for the next property, and shall be used also later on. Given a reduction relation $\to$, the predicate $t$ *is* $SN_\to$, also noted $t \in SN_\to$, is defined *inductively* as follows:

- $t$ is $SN_\to$ if $t$ is $\to$-normal, i.e., has no $\to$ redexes.
- $t$ is $SN_\to$ if $s$ is $SN_\to$ for all $s$ such that $t \to s$.

We use $SN_\to$ for the set of terms that are $SN_\to$, and say that $\to$ is strongly normalizing (or $SN$) if $t \in SN_\to$ for all terms $t$.

**Local Termination.** While terms in the LSC can be divergent, because the LSC simulates the $\lambda$-calculus, an important property is that the rewriting rules of the LSC are strongly normalizing when considered separately.

**Proposition 4.1** (Local termination, [Acc12])**.** *Let* $\to \in \{\to_{\mathtt{dB}}, \to_{\mathtt{ls}}, \to_{\mathtt{gc}}\}$*. Then* $\to$ *is strongly normalizing.*

The only non-trivial point is the strong normalization of $\to_{\mathtt{ls}}$, which requires a measure. It easily extends to the one of $\to_{\mathtt{ls}} \cup \to_{\mathtt{gc}}$, that is, of the sub-system rewriting ES. The ESC shall preserve this key property, as we shall prove in Sect. 8.

**Structural Equivalence.** The LSC is often enriched with the following equivalence.

**Definition 4.2** (Structural equivalence)**.** Define head contexts as follows:
$$\text{Head ctxs } H \ ::= \ \langle\cdot\rangle \mid Ht \mid \lambda x.H \mid H[x{\leftarrow}s]$$
If $x \notin \mathtt{fv}(H)$ and $H$ does not capture variables in $\mathtt{fv}(s)$, set:
$$H\langle t\rangle[x{\leftarrow}s] \quad \sim \quad H\langle t[x{\leftarrow}s]\rangle$$
*Structural equivalence* $\equiv$ is the closure of $\sim$ by reflexivity, symmetry, transitivity, and contexts $C$.

Its key property is that it commutes with evaluation in the following strong sense.

**Proposition 4.3** ($\equiv$ is a strong bisimulation wrt $\to_{LSC}$ [ABKL14])**.** *Let* $a \in \{\mathtt{dB}, \mathtt{ls}, \mathtt{gc}\}$*. If* $t \equiv s \to_a u$ *then exists* $r$ *such that* $t \to_a r \equiv u$*.*

Essentially, $\equiv$ never creates redexes, it can be postponed, and vanishes on normal forms (that have no ES). Accattoli shows that $\equiv$ is exactly the quotient induced by the (call-by-name) translation of $\lambda$-terms with ES to proof nets [Acc18].

**Linear Head Reduction.** One of the key properties of the LSC is that it admits a neat definition of Danos & Regnier's and Mascari & Pedicini's linear head reduction [MP94, DR04]. Before the introduction of the LSC, the presentations of linear head reduction were instead hard to manage. To define it, we need the notion of head context:
$$\text{Head contexts } \quad H \ ::= \ \langle\cdot\rangle \mid Ht \mid \lambda x.H \mid H[x{\leftarrow}t]$$
Then head contexts are first used to define the root linear head substitution rule:
$$\text{Linear Head Substitution} \quad H\langle\!\langle x\rangle\!\rangle[x{\leftarrow}s] \ \mapsto_{\mathtt{lhs}} \ H\langle\!\langle s\rangle\!\rangle[x{\leftarrow}s]$$
Finally, the root rules $\mapsto_{\mathtt{dB}}$, $\mapsto_{\mathtt{lhs}}$, and $\mapsto_{\mathtt{gc}}$ are closed by *head* contexts:

$$\text{Contextual closures}$$

$$\frac{t \mapsto_{\mathtt{dB}} t'}{H\langle t\rangle \to_{\mathtt{lhdB}} H\langle t'\rangle} \qquad \frac{t \mapsto_{\mathtt{lhs}} t'}{H\langle t\rangle \to_{\mathtt{lhs}} H\langle t'\rangle} \qquad \frac{t \mapsto_{\mathtt{gc}} t'}{H\langle t\rangle \to_{\mathtt{lhgc}} H\langle t'\rangle}$$

$$\text{Linear head reduction} \quad \to_{\mathtt{lh}} \ :=\, \to_{\mathtt{lhdB}} \cup \to_{\mathtt{lhs}} \cup \to_{\mathtt{lhgc}}$$

Linear head reduction has many interesting property. For the present paper, the most relevant one is the *sub-term property*, which, for the given presentation, is proved by Accattoli and Dal Lago in [ADL12] (it was already used by Danos & Regnier's and Mascari & Pedicini, but not exploited for complexity analyses).

**Theorem 4.4** (Sub-term property for $\to_{\mathtt{lh}}$, [ADL12])**.** *All the sub-terms duplicated by* $\to_{\mathtt{lhs}}$ *along a linear head evaluation sequence* $t \to_{\mathtt{lh}}^* s$ *are sub-term of* $t$ *(up to variables renaming).*

The good strategy for IMELL that we shall define in Sect. 11 generalizes linear head reduction to the whole of IMELL and preserves its sub-term property. For that, note that $\to_{\mathtt{lh}}$ never evaluates inside ES. Similarly, the good strategy shall not evaluate inside cuts. Note also that $\to_{\mathtt{lh}}$ does not compute normal forms, as it never evaluates arguments. Since we want the good strategy to compute cut-free proofs, we shall have to evaluate the analogous of arguments in our framework (namely, the left premises of subtractions). This shall be obtained by entering into arguments only under some conditions, similarly to how $\to_{\mathtt{lh}}$ is generalized to enter inside arguments by Accattoli and co-authors [ADL16, ABM15], obtaining linear leftmost-outermost reduction.

## 5. Towards Exponentials as Substitutions

This section is an informal and hopefully intuitive introduction to our cut elimination for IMELL. The logical rules we refer to are standard (but decorated with terms), see Fig. 3.

**Splitting Terms.** For the moment, we prefer to avoid giving grammars, but we want nonetheless to fix some terminologies and intuitions. We distinguish variables into multiplicatives, noted $m, n, o$, and exponentials, noted $e, f, g$, and use $x, y, z$ for variables of unspecified kind. Proof terms for axioms and right introduction rules shall be called *values*, and noted $v$. We borrow from the LSC the notation $L$ (here standing for *L*eft context) and generalize it to a context corresponding to (the term annotations induced by) a sequence of left rules of the sequent calculus. Exactly as IMELL proofs can be seen as ending on a sequence of left rules following a right rule or an axiom, every proof term $t$ shall uniquely (on-the-fly) decompose as $L\langle v\rangle$, called its *splitting*.

**Meta-Level Substitution.** For the design of our system, the first step is understanding the subtle interplay between linearity and substitution. Consider a cut on an exponential formula, and a first attempt at decorating it with terms:

$$\frac{\Gamma \vdash s : !A \qquad \Delta, e : !A \vdash t : B}{\Gamma, \Delta \vdash [s \to e]t : B} \ \mathtt{cut}$$

Note that the notation $[s \to e]t$ for cuts is similar but opposite to the one for ES in the LSC, as the arrow goes from left to right and it binds on the right (in $t$), to better reflect the structure of sequent proofs. We seek a notion of meta-level substitution $\{s \to e\}t$ and a small-step rule such as:

$$[s \to e]t \ \to \ \{s \to e\}t \tag{5.1}$$

Perhaps surprisingly, linear logic does not validate this rule. Despite the $!A$ type, not all of $s$, indeed, might be allowed to be duplicated. Let's refine the example and consider the case in which the last rule contributing to $s$ is, say, the left rule for $\otimes$, what we shall call a *par* and annotate with $(\!(m \mathbin{\gamma} x, y)\!)$:

$$\frac{\dfrac{\Gamma, x : B, y : C \vdash s' : !A}{\Gamma, m : B \otimes C \vdash (\!(m \mathbin{\gamma} x, y)\!) s' : !A} \ \otimes_l \qquad \Delta, e : !A \vdash t : B}{\Gamma, m : B \otimes C, \Delta \vdash [(\!(m \mathbin{\gamma} x, y)\!) s' \to e]t : B} \ \mathtt{cut}$$

Now we can see why (5.1) is not valid: the annotation $(\!(m \mathbin{\gamma} x, y)\!)$ being linear, it should not be duplicated. The proof nets approach to this issue is marking with a box the sub-term/proof net of $s'$ that can be duplicated. The usual sequent calculus approach, instead, is commuting the cut with $\otimes_l$, which on proof terms corresponds to have a rule such as:

$$[(\!| m \,\mathfrak{N}\, x, y |\!) s' \to e] t \to (\!| m \,\mathfrak{N}\, x, y |\!) [s' \to e] t$$

and then keep commuting the cut with left rules until an introduction of a ! connective on the right—what we call *an exponential value*, noted $v_\mathsf{e}$—is reached, which can then trigger the substitution. But these commutative rules are a burden, and exactly what the LSC philosophy rejects.

The idea is to exploit the splitting of terms into values and left contexts, so as to import on terms the proof nets approach. A term $s$ of exponential type, in fact, shall always be splittable as $L\langle v_\mathsf{e}\rangle$, that is, as a left context and an exponential value. We then refine (5.1) as follows:

$$[s \to e] t = [L\langle v_\mathsf{e}\rangle \to e] t \to L\langle \{v_\mathsf{e} \to e\} t\rangle$$

Meta-level substitution thus concerns only exponential values $v_\mathsf{e}$—playing the role of proof nets boxes—and *not* the surrounding left context $L$. This same mechanism is used in Accattoli and Paolini's *value substitution calculus* [AP12].

**Split Cuts.** In fact, we adopt a further refinement. We restrict cuts to have only values as left sub-terms, that is, of being of shape $[v \to x] t$. Since we want to stick to the standard sequent calculus for IMELL, we do not restrict the typing rule, we do instead modify its decoration:

$$\frac{\Gamma \vdash s = L\langle v\rangle : !A \qquad \Delta, e : !A \vdash t : B}{\Gamma, \Delta \vdash L\langle [v \to e] t\rangle : B} \; \mathtt{cut}$$

And we shall do the same for the left rule $\multimap_l$ for $\multimap$ (see Fig. 3). This refinement simplifies various technical points, in particular it enables a simple definition of the good strategy.

**Micro-Step Substitution.** For micro-step substitution we simply borrow the LSC approach, adopting rules such as:

$$[v_\mathsf{e} \to e] C \langle\!\langle e \rangle\!\rangle \to [v_\mathsf{e} \to e] C \langle\!\langle v_\mathsf{e} \rangle\!\rangle$$

While in line with the LSC, this is *not* how cut elimination is usually performed in both proof nets and the sequent calculus. There are three main differences:

(1) *LSC-style duplications*: in our case duplication happens also when there are no contractions involved, for instance we shall have $[v_\mathsf{e} \to e] e \to [v_\mathsf{e} \to e] v_\mathsf{e}$, and there shall be a rule $(\to_!)$ duplicating and interacting with derelictions at the same time (see the next section). Not only this approach is perfectly sound, it actually has better rewriting properties than the traditional one.

(2) *No commutations*: there are no commutations with !. Even in proof nets, where most commutative cuts vanish, micro-step cut elimination usually relies on a commutative rule for !-boxes (namely $\to_\boxdot$ in Fig. 1), that on proof terms would look as follows (when $e$ occurs in $t$):

$$[v_\mathsf{e} \to e] ! t \to ! [v_\mathsf{e} \to e] t$$

The LSC philosophy rejects such a commutative rule.

(3) *Axioms*: in the proof nets literature, cut elimination of exponential axioms is identical to the one for multiplicative axioms (there is in fact no distinction between the two kinds of axioms) and simply amounts to the removal of the cut and the axiom, in both the small-step (*à la* Regnier) and the micro-step (*à la* Girard) approaches. In (asymmetric) intuitionistic settings, axioms have two rules, depending on whether they are substituted, or substituted upon. In our syntax, the traditional approach to cut elimination for exponential axioms would take the following form:

$$
\begin{array}{rrcl}
\textsc{Mul. vars} & m, n, o & \in & \mathcal{V}_{\mathtt{m}} \\
\textsc{Exp. vars} & e, f, g & \in & \mathcal{V}_{\mathtt{e}} \\
\textsc{Variables} & x, y, z & \in & \mathcal{V} := \mathcal{V}_{\mathtt{m}} \uplus \mathcal{V}_{\mathtt{e}} \\
\\
\textsc{Mul. values} & v_{\mathtt{m}} & ::= & m \in \mathcal{V}_{\mathtt{m}} \mid (t, s) \mid \lambda x.t \\
\textsc{Exp. values} & v_{\mathtt{e}} & ::= & e \in \mathcal{V}_{\mathtt{e}} \mid {!}t \\
\textsc{Values} & v & ::= & v_{\mathtt{m}} \mid v_{\mathtt{e}} \\
\\
\textsc{Terms} \quad t, s, u & & ::= & v \mid [v{\to}x]t \mid (\!|m\,\mathfrak{N}\,x, y|\!)t \mid (\!|m\,\otimes\,v, x|\!)t \mid (\!|e?x|\!)t \\
\\
\textsc{Left ctxs} & L & ::= & \langle\cdot\rangle \mid [v{\to}x]L \mid (\!|m\,\mathfrak{N}\,x, y|\!)L \mid (\!|m\,\otimes\,v, x|\!)L \mid (\!|e?x|\!)L \\
\textsc{Value ctxs} & V & ::= & \langle\cdot\rangle \mid (C, s) \mid (t, C) \mid \lambda x.C \mid {!}C \\
\textsc{General ctxs} & C & ::= & V \mid [V{\to}x]t \mid (\!|m\,\otimes\,V, x|\!)t \mid L\langle C\rangle \\
\textsc{Mul. value ctxs} & V_{\mathtt{m}} & ::= & \langle\cdot\rangle \mid (M, s) \mid (t, M) \mid \lambda x.M \\
\textsc{Mul. ctxs} & M & ::= & V_{\mathtt{m}} \mid [V_{\mathtt{m}}{\to}x]t \mid (\!|m\,\otimes\,V_{\mathtt{m}}, x|\!)t \mid L\langle M\rangle
\end{array}
$$

Figure 2: Grammars of the exponential substitution calculus.

$$[f{\to}e]t \quad \to \quad \{f{\to}e\}t$$

When written with terms, one immediately notices that such a rule is small-step. In Sect. 10, we shall show that such a small-step rule generates a closable but unpleasant local confluence diagram with the other micro-step exponential rules, which is avoided by adopting a LSC-duplication style also for exponential axioms/variables.

## 6. The Exponential Substitution Calculus

**Values and Terms.** The grammars of the ESC are in Fig. 2. First of all, a disclaimer. The constructors of the calculus are an untyped *minimalist encoding* of the sequent calculus rules, and *not* an intuitively readable calculus. For that aim, one would rather use `let-in` notations. Variables are of two disjoint kinds, multiplicative and exponential, and we often refer to variables of unspecified kind using the notations $x, y, z$. Values are the proof terms associated to axioms or to the right rules and, beyond variables, are *abstractions* $\lambda x.t$, *tensor pairs* $(t, s)$, and *promotions* ${!}t$. The proof terms decorating left rules are *pars* $(\!|m\,\mathfrak{N}\,x, y|\!)t$, *subtractions* $(\!|m\,\otimes\,v, x|\!)t$, *derelictions* $(\!|e?x|\!)t$, and *cuts* $[v{\to}x]t$, which is in red because of its special role. Note that cuts and subtractions are *split*, that is, have values (rather than terms) as left sub-terms. Back to readability, subtraction might also be decorated with `let` $x = mv$ `in` $t$, but note that this notation is less faithful to the logical rule because the two premises are represented asymmetrically, and that it uses an application constructor that rather belongs to natural deduction.

In $\lambda x.t$, $(\!|m\,\otimes\,v, x|\!)t$, $(\!|e?x|\!)t$, and $[v{\to}x]t$ we have that $x$ is bound in $t$, and in $(\!|m\,\mathfrak{N}\,x, y|\!)t$ both $x$ and $y$ are bound in $t$. We identify terms up to $\alpha$-renaming. Free variables (resp. multiplicative/exponential variables) are defined as expected, and noted $\mathtt{fv}(t)$ (resp. $\mathtt{mfv}(t)$ and $\mathtt{efv}(t)$). We use $|t|_x$ for the number of free occurrences of $x$ in $t$. There is a notion of *proper term* ensuring the linearity of multiplicative variables. The only relevant case is: ${!}t$ *is proper if $t$ is proper and* $\mathtt{mfv}(t) = \emptyset$.

**Definition 6.1** (Proper terms). Proper terms are defined by induction on $t$ as follows:

- *Variables*: $m$ and $e$ are proper.
- *Tensor*: $(t, s)$ is proper if $t$ is proper, $s$ is proper, and $\mathtt{mfv}(t) \cap \mathtt{mfv}(s) = \emptyset$
- *Par*: $(\!|m \mathbin{\bindnasrepma} x, y|\!)t$ is proper if $t$ is proper and $m \notin (\mathtt{mfv}(t) \setminus x, y)$
- *Implication*: $\lambda x.t$ is proper if $t$ is proper and if $x$ is a multiplicative variable then $x \in \mathtt{mfv}(t)$.
- *Subtraction*: $(\!|m \oslash v, x|\!)t$ is proper if $t$ is proper, $v$ is proper, $\mathtt{mfv}(v) \cap (\mathtt{mfv}(t) \setminus \{x\}) = \emptyset$, $m \notin \mathtt{mfv}(t)$, and if $x$ is a multiplicative variable then $x \in \mathtt{mfv}(t)$.
- *Bang*: $!t$ is proper if $t$ is proper and $\mathtt{mfv}(t) = \emptyset$.
- *Dereliction*: $(\!|e?x|\!)t$ is proper if $t$ is proper and if $x$ is a multiplicative variable then $x \in \mathtt{mfv}(t)$.
- *Cut*: $[v{\to}x]t$ is proper if $t$ is proper, $v$ is proper, $\mathtt{mfv}(v) \cap (\mathtt{mfv}(t) \setminus \{x\}) = \emptyset$, and if $x$ is a multiplicative variable then $x \in \mathtt{mfv}(t)$.

**Typing System.** The typing rules are in Fig. 3. The formulas of IMELL and the deductive rules of the sequent calculus are standard, but for the decoration with proof terms and the side conditions about variable names. Linear implication $\multimap$ is also referred to as *lolli*. The only atomic formula that we consider, $X_\mathtt{m}$, is multiplicative. There is no multiplicative unit because in presence of the exponentials 1 can be simulated by $!(X_\mathtt{m} \multimap X_\mathtt{m})$. We distinguish between multiplicative and exponential axioms, in order to decorate them with the corresponding kind of variable. Note that the weakening and contraction rules do not add constructors to terms. This is crucial in order to keep the calculus manageable. Note also that the decorations of the cut and $\multimap_l$ rules are *split*, as explained in the previous section. Clearly, typed terms are proper. We use $\pi \triangleright \Gamma \vdash t : A$ for a proof/typing derivation $\pi$ ending in that sequent.

**Contexts and Plugging.** The broadest notion of context that we consider is *general contexts* $C$, which simply allow the hole $\langle \cdot \rangle$ to replace any sub-term in a term. Because of split cuts and subtractions, the definition relies on the auxiliary notion of *value context* $V$. The definition also uses *left contexts* $L$, which are contexts under left constructors (or, for binary left constructors, under the right sub-term) that play a key role in the system—their use in defining $C$ is just to keep the grammar compact (to fit it in the figure).

A fact used pervasively is that every term $t$ writes, or *splits* uniquely as $t = L\langle v \rangle$. For instance $(\!|e?m|\!)[v{\to}n](m, n)$ splits as $L = (\!|e?m|\!)[v{\to}n]\langle \cdot \rangle$ and $v = (m, n)$. We also have *multiplicative contexts* $M$, that the proof nets literature would call *level 0* contexts, that is, contexts with the hole out of all !, and which also rely on their own value contexts $V_\mathtt{m}$.

Because of split cuts and subtractions, plugging is slightly tricky, as it has to preserve the split shape. For instance, we have $([\langle \cdot \rangle{\to}o]t)\langle (\!|e?m|\!)(m, n) \rangle = (\!|e?m|\!)[(m, n){\to}o]t$. A similar approach to plugging is also used by Accattoli et al. in [ACGSC19]. The full definition follows. It is mostly as expected, the only two subtle cases are for $[\langle \cdot \rangle{\to}x]t$ and $(\!|m \oslash \langle \cdot \rangle, x|\!)$.

**Definition 6.2** (Plugging). Plugging $C\langle t \rangle$ is defined as follows:

$$\text{Formulas} \quad A, B, C \quad ::= \quad X_{\mathtt{m}} \mid A \otimes B \mid A \multimap B \mid {!}A$$

### Multiplicative right rules

$$\frac{A \neq {!}B}{m : A \vdash m : A} \ \mathtt{ax_m}$$

$$\frac{\Gamma \vdash t : A \qquad \Delta \vdash s : B \qquad \Gamma \# \Delta}{\Gamma, \Delta \vdash (t, s) : A \otimes B} \ \otimes_r$$

$$\frac{x : A, \Gamma \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B} \ \multimap_r$$

### Multiplicative left rules

$$\frac{\Gamma \vdash L\langle v \rangle : A \qquad \Delta, x : A \vdash t : B \qquad \Gamma \#(\Delta, x : A)}{\Gamma, \Delta \vdash L\langle [v \to x] t \rangle : B} \ \mathtt{cut}$$

$$\frac{\Gamma, x : A, y : B \vdash t : C \qquad m \text{ fresh}}{\Gamma, m : A \otimes B \vdash (\!|m^{28}x, y|\!) t : C} \ \otimes_l$$

$$\frac{\Gamma \vdash L\langle v \rangle : A \qquad \Delta, x : B \vdash t : C \qquad \Gamma \#(\Delta, x : B), \ m \text{ fresh}}{\Gamma, \Delta, m : A \multimap B \vdash L\langle (\!|m \oslash v, x|\!) t \rangle : C} \ \multimap_l$$

### Exponential rules

$$\frac{}{e : {!}A \vdash e : {!}A} \ \mathtt{ax_e}$$

$$\frac{\Gamma, x : A \vdash t : B \qquad e \text{ fresh}}{\Gamma, e : {!}A \vdash (\!|e?x|\!) t : B} \ {!}_l \qquad\qquad \frac{{!}\Gamma \vdash t : A}{{!}\Gamma \vdash {!}t : {!}A} \ {!}_r$$

$$\frac{\Gamma \vdash t : A \qquad e \text{ fresh}}{\Gamma, e : {!}B \vdash t : A} \ \mathtt{w} \qquad\qquad \frac{\Gamma, e : {!}B, f : {!}B \vdash t : A}{\Gamma, e : {!}B \vdash \{e \to f\} t : A} \ \mathtt{c}$$

Figure 3: IMELL typing rules for the ESC, where $\Gamma \# \Delta$ is a shortcut for $\mathtt{dom}(\Gamma) \cap \mathtt{dom}(\Delta) = \emptyset$.

### Plugging

$$
\begin{aligned}
\langle \cdot \rangle \langle t \rangle &:= t & (\lambda x.C)\langle t \rangle &:= \lambda x.C\langle t \rangle \\
(C, s)\langle t \rangle &:= (C\langle t \rangle, s) & (s, C)\langle t \rangle &:= (s, C\langle t \rangle) \\
({!}C)\langle t \rangle &:= {!}C\langle t \rangle & ((\!|e?x|\!)C)\langle t \rangle &:= (\!|e?x|\!)C\langle t \rangle \\
((\!|m^{28}x, y|\!)C)\langle t \rangle &:= (\!|m^{28}x, y|\!)C\langle t \rangle \quad & ((\!|m \oslash v, x|\!)C)\langle t \rangle &:= (\!|m \oslash v, x|\!)C\langle t \rangle \\
([v \to x]C)\langle t \rangle &:= [v \to x]C\langle t \rangle
\end{aligned}
$$

$$((\!|m \oslash V, x|\!)s)\langle t \rangle := \begin{cases} L\langle (\!|m \oslash v, x|\!)s \rangle & \text{if } V = \langle \cdot \rangle \text{ and } t = L\langle v \rangle \\ (\!|m \oslash V\langle t \rangle, x|\!)s & \text{otherwise} \end{cases}$$

$$([V \to x]s)\langle t \rangle := \begin{cases} L\langle [v \to x]s \rangle & \text{if } V = \langle \cdot \rangle \text{ and } t = L\langle v \rangle \\ [V\langle t \rangle \to x]s & \text{otherwise} \end{cases}$$

Plugging $C\langle C'\rangle$ of contexts is defined similarly, and plugging is extended to all forms of contexts by seeing them as general contexts. As for the LSC, plugging can capture variables and we use $C\langle\!\langle t\rangle\!\rangle$ when we want to prevent it.

With our notion of plugging, a left constructor in a term might be identified via plugging of a term in a context in various ways, e.g. for $t := (\!(e?m)\![m{\to}n]n, f)$ we have $t = C\langle(\!(e?m)\!)m\rangle$ with $C := ([\langle\cdot\rangle{\to}n]n, f)$, and $t = C'\langle(\!(e?m)\![m{\to}n]n\rangle$ with $C' := (\langle\cdot\rangle, f)$. For most uses, in particular for defining the rewriting rules, this is harmless. It shall play a role, however, for defining the good strategy, because it shall be important whether the hole of the context is inside or outside a cut. Then note that in the first decomposition the term $(\!(e?m)\!)m$ is not a sub-term of $t$. This suggest the following notion. A *position* in a term $t$ is a decomposition $t = C\langle s\rangle$ such that $s$ is a sub-term of $t$ (as for $C'\langle(\!(e?m)\![m{\to}n]n\rangle$ above).

We can now state the property ensured by properness.

**Lemma 6.3** (Structural linearity). *Let $t$ be a proper term and $m \in \mathtt{mfv}(t)$. Then $|t|_m = 1$ and $t = M\langle\!\langle m\rangle\!\rangle$ for some $M$.*

*Proof.* By induction on $t$. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\square$

**Meta-Level Substitution and the Small-Step Exponential Rule.** The rewriting rules are in Fig. 4. For defining the small-step exponential cut elimination we need meta-level substitution $\{v_\mathsf{e}{\to}e\}t$, which is defined only for exponential values (that is, exponential variables $f$ and promotions $!t$). The long definition is mostly as expected, but for one case, explained after the definition.

**Definition 6.4** (Meta-level exponential substitution). The meta-level (exponential) substitution $\{v_\mathsf{e}{\to}e\}t$ of the exponential value $v_\mathsf{e}$ for the free occurrences of $e$ in $t$ is defined by induction on $t$ as follows (assuming on-the-fly $\alpha$-renaming to avoid capture of free variables, omitted for ease of reading and manipulation):

<div align="center">

META-LEVEL SUBSTITUTION

</div>

$$
\begin{aligned}
\{v_\mathsf{e}{\to}e\}m &:= m & \{v_\mathsf{e}{\to}e\}(\!(m\,\mathfrak{N}\,x, y)\!)t &:= (\!(m\,\mathfrak{N}\,x, y)\!)\{v_\mathsf{e}{\to}e\}t \\
\{v_\mathsf{e}{\to}e\}f &:= f & \{v_\mathsf{e}{\to}e\}(\!(m{\otimes}v, x)\!)t &:= (\!(m{\otimes}\{v_\mathsf{e}{\to}e\}v, x)\!)\{v_\mathsf{e}{\to}e\}t \\
\{v_\mathsf{e}{\to}e\}e &:= v_\mathsf{e} & \{v_\mathsf{e}{\to}e\}(\!(f?x)\!)t &:= (\!(f?x)\!)\{v_\mathsf{e}{\to}e\}t \\
\{v_\mathsf{e}{\to}e\}(t, u) &:= (\{v_\mathsf{e}{\to}e\}t, \{v_\mathsf{e}{\to}e\}u) & \{f{\to}e\}(\!(e?x)\!)t &:= (\!(f?x)\!)\{v_\mathsf{e}{\to}e\}t \\
\{v_\mathsf{e}{\to}e\}\lambda x.t &:= \lambda x.\{v_\mathsf{e}{\to}e\}t & \{!L\langle v\rangle{\to}e\}(\!(e?x)\!)t &:= L\langle[v{\to}x]\{v_\mathsf{e}{\to}e\}t\rangle \\
\{v_\mathsf{e}{\to}e\}!t &:= !\{v_\mathsf{e}{\to}e\}t & \{v_\mathsf{e}{\to}e\}[v{\to}x]t &:= [\{v_\mathsf{e}{\to}e\}v{\to}x]\{v_\mathsf{e}{\to}e\}t
\end{aligned}
$$

The key case is the substitution of a promotion for a dereliction (the second to last in the right column), that removes the dereliction, copies the promotion body, and puts it in a cut—in proof nets jargon, *it opens the box*. To preserve the split shape, the body of the promotion is split and only the value is cut. An example:

$$\{![f{\to}g]g{\to}e\}(\!(e?e')\!)(e', e) = [f{\to}g][g{\to}e'](e', ![f{\to}g]g)$$

Our definition verifies the expected properties of meta-level substitution. Some are given by the next two lemmas.

**Lemma 6.5** (Basic properties of meta-level substitution).

(1) *If $v_\mathsf{e}$ and $t$ are proper then $\{v_\mathsf{e}{\to}e\}t$ is proper.*
(2) *$\{v_\mathsf{e}{\to}e\}v'_\mathsf{e}$ is an exponential value and $\{v_\mathsf{e}{\to}e\}v_\mathsf{m}$ is a multiplicatie value.*
(3) *If $C$ does not capture free variables of $v_\mathsf{e}$ then $\{v_\mathsf{e}{\to}e\}C\langle t\rangle = (\{v_\mathsf{e}{\to}e\}C)\langle\{v_\mathsf{e}{\to}e\}t\rangle$.*

$$\boxed{\begin{array}{c}
\textsc{Root multiplicative rules} \\[1em]
\begin{array}{rcl}
[v_{\mathtt{m}}{\to}m]M\langle\!\langle m\rangle\!\rangle & \mapsto_{\mathtt{axm}_1} & M\langle\!\langle v_{\mathtt{m}}\rangle\!\rangle \\
[n{\to}m]t & \mapsto_{\mathtt{axm}_2} & \{n{\to}m\}t \\
[(s,u){\to}m]M\langle(\!|m\,\mathcal{R}\,x,y|\!)t\rangle & \mapsto_{\otimes} & M\langle L\langle[v{\to}x]L'\langle[v'{\to}y]t\rangle\rangle\rangle \\
& & \text{with } s = L\langle v\rangle \text{ and } u = L'\langle v'\rangle \\[0.8em]
[\lambda y.s{\to}m]M\langle(\!|m\oslash v,x|\!)t\rangle & \mapsto_{\multimap} & M\langle[v{\to}y]L\langle[v'{\to}x]t\rangle\rangle \\
& & \text{with } s = L\langle v'\rangle
\end{array} \\[2em]
\hline \\
\textsc{Micro-step root exponential rules} \\[1em]
\begin{array}{rclrcl}
[v_{\mathtt{e}}{\to}e]C\langle\!\langle e\rangle\!\rangle & \mapsto_{\mathtt{axe}_1} & & [v_{\mathtt{e}}{\to}e]C\langle\!\langle v_{\mathtt{e}}\rangle\!\rangle \\
[f{\to}e]C\langle(\!|e?x|\!)t\rangle & \mapsto_{\mathtt{axe}_2} & & [f{\to}e]C\langle(\!|f?x|\!)t\rangle \\
[!s{\to}e]C\langle(\!|e?x|\!)t\rangle & \mapsto_{!} & & [!s{\to}e]C\langle L\langle[v{\to}x]t\rangle\rangle \\
& & & \text{with } s = L\langle v\rangle \\[0.8em]
[v_{\mathtt{e}}{\to}e]t & \mapsto_{\mathtt{w}} & & t \quad \text{if } e \notin \mathtt{fv}(t)
\end{array} \\[2em]
\hline \\
\textsc{Small-step root exponential rule} \\[1em]
[v_{\mathtt{e}}{\to}e]t \quad \mapsto_{\mathtt{e^{ss}}} \quad \{v_{\mathtt{e}}{\to}e\}t \\[2em]
\hline \\
\textsc{Contextual closure} \\[1em]
\dfrac{t \mapsto_a s}{C\langle t\rangle \to_a C\langle s\rangle} \quad \text{for } a \in \left\{\begin{array}{l}\mathtt{axm}_1, \mathtt{axm}_2, \otimes, \multimap, \\ \mathtt{axe}_1, \mathtt{axe}_2, !, \mathtt{w}, \mathtt{e^{ss}}\end{array}\right\} \\[2em]
\hline \\
\textsc{Notations} \\[0.5em]
\begin{array}{rrcl}
\textsc{Multiplicative} & \to_{\mathtt{m}} & := & \to_{\mathtt{axm}_1} \cup \to_{\mathtt{axm}_2} \cup \to_{\otimes} \cup \to_{\multimap} \\
\textsc{Exponential micro-step} & \to_{\mathtt{e^{ms}}} & := & \to_{\mathtt{axe}_1} \cup \to_{\mathtt{axe}_2} \cup \to_{!} \cup \to_{\mathtt{w}} \\
\textsc{Small-step} & \to_{\mathtt{ss}} & := & \to_{\mathtt{m}} \cup \to_{\mathtt{e^{ss}}} \\
\textsc{Micro-step} & \to_{\mathtt{ms}} & := & \to_{\mathtt{m}} \cup \to_{\mathtt{e^{ms}}} \\
\textsc{Non-erasing micro-step} & \to_{\mathtt{ms}_{\neg\mathtt{w}}} & := & \to_{\mathtt{m}} \cup \to_{\mathtt{axe}_1} \cup \to_{\mathtt{axe}_2} \cup \to_{!} \\
\textsc{Non-lolli micro-step} & \to_{\mathtt{ms}_{\neg\multimap}} & := & \to_{\mathtt{axm}_1} \cup \to_{\mathtt{axm}_2} \cup \to_{\otimes} \cup \to_{\mathtt{e^{ms}}} \\
\textsc{Non-lolli small-step} & \to_{\mathtt{ss}_{\neg\multimap}} & := & \to_{\mathtt{axm}_1} \cup \to_{\mathtt{axm}_2} \cup \to_{\otimes} \cup \to_{\mathtt{e^{ss}}}
\end{array}
\end{array}}$$

Figure 4: Rewriting rules of the ESC.

(4) $\{v_{\mathtt{e}}{\to}e\}s = s$ *if $e \notin \mathtt{fv}(s)$.*

(5) $\{v_{\mathtt{e}}{\to}f\}\{v'_{\mathtt{e}}{\to}e\}t = \{\{v_{\mathtt{e}}{\to}f\}v'_{\mathtt{e}}{\to}e\}\{v_{\mathtt{e}}{\to}f\}t.$

(6) $\{v_{\mathtt{e}}{\to}e\}\{v'_{\mathtt{e}}{\to}f\}u = \{v'_{\mathtt{e}}{\to}f\}\{v_{\mathtt{e}}{\to}e\}u$ *if $e \notin \mathtt{fv}(s)$ and $e \notin \mathtt{fv}(u)$.*

(7) $\{v_{\mathtt{e}}{\to}e\}\{n{\to}m\}u = \{n{\to}m\}\{v_{\mathtt{e}}{\to}e\}u.$

(8) $\{!u{\to}e\}\{!u{\to}f\}t = \{!u{\to}e\}\{e{\to}f\}t.$

**Lemma 6.6** (Stability of steps under substitution)**.**

(1) *If $t \to_a s$ then $\{v_{\mathtt{e}}{\to}e\}t \to_a \{v_{\mathtt{e}}{\to}e\}s$ for $a \in \{\mathtt{axm}_1, \mathtt{axm}_2, \otimes, \multimap, \mathtt{e^{ss}}, \mathtt{e^{ms}}\}.$*

(2) *If $v_{\mathsf{e}} \to_a v_{\mathsf{e}}'$ then $\{v_{\mathsf{e}}{\to}e\}t \to_a^* \{v_{\mathsf{e}}'{\to}e\}t$ for $a \in \{\mathtt{axm}_1, \mathtt{axm}_2, \otimes, \multimap, \mathsf{e}^{\mathsf{ss}}, \mathtt{axe}_1, \mathtt{axe}_2, !, \mathtt{w}\}$.*

*Proof.*

(1) By induction on $t \to_a s$. Details are in [Acc22]. The only subtle case is the one for $\mapsto_{\mathtt{axe}_2}$, which has a sub-case where the step might become a $\mapsto_!$ step after substitution. Let $t = [g{\to}f]C\langle\!(\!|f?x|\!)u\rangle \mapsto_{\mathtt{axe}_2} [g{\to}f]C\langle\!(\!|g?x|\!)u\rangle = s$, $g = e$, and $v_{\mathsf{e}} = !L\langle v\rangle$.. Then:

$$
\begin{aligned}
\{v_{\mathsf{e}}{\to}g\}t &= & \{v_{\mathsf{e}}{\to}g\}[g{\to}f]C\langle\!(\!|f?x|\!)u\rangle \\
&= & [v_{\mathsf{e}}{\to}f]\{v_{\mathsf{e}}{\to}g\}C\langle\!(\!|f?x|\!)u\rangle \\
&=_{L.6.5.3} & [v_{\mathsf{e}}{\to}f](\{v_{\mathsf{e}}{\to}g\}C)\langle(\!|f?x|\!)\{v_{\mathsf{e}}{\to}g\}u\rangle \\
&\mapsto_! & [v_{\mathsf{e}}{\to}f](\{v_{\mathsf{e}}{\to}g\}C)\langle L\langle[v{\to}x]\{v_{\mathsf{e}}{\to}g\}u\rangle\rangle \\
&= & [v_{\mathsf{e}}{\to}f](\{v_{\mathsf{e}}{\to}g\}C)\langle\{v_{\mathsf{e}}{\to}g\}(\!|g?x|\!)u\rangle \\
&=_{L.6.5.3} & [v_{\mathsf{e}}{\to}f]\{v_{\mathsf{e}}{\to}g\}C\langle\!(\!|g?x|\!)u\rangle \\
&= & \{v_{\mathsf{e}}{\to}g\}[g{\to}f]C\langle\!(\!|g?x|\!)u\rangle & = & \{v_{\mathsf{e}}{\to}g\}s
\end{aligned}
$$

(2) By induction on $t$.     □

The small-step exponential rule $\to_{\mathsf{e}^{\mathsf{ss}}}$ then simply turns a cut $[v_{\mathsf{e}}{\to}e]t$ into the corresponding meta-level substitution.

**Multiplicative Cut Elimination Rules.** The ESC has four multiplicative rules, in particular two for axioms, depending on whether they are acted upon ($\to_{\mathtt{axm}_1}$) or used to rename another multiplicative (thus linear) variable ($\to_{\mathtt{axm}_2}$). For $\to_{\mathtt{axm}_2}$, we abuse notations and use a form of meta-level substitution $\{n{\to}m\}t$ for the renaming operation. There is also a slight superposition between $\to_{\mathtt{axm}_1}$ and $\to_{\mathtt{axm}_2}$, for instance in $[m{\to}n]\lambda e.n \to \lambda e.m$, which can be both kinds of steps. To disambiguate, one should refine $\to_{\mathtt{axm}_2}$ into two rules:

$$
\begin{aligned}
[n{\to}m]M\langle\!(\!|m\,\mathfrak{V}x, y|\!)t\rangle &\to_{\mathtt{axm}_2'} & M\langle\!(\!|n\,\mathfrak{V}x, y|\!)t\rangle \\
[n{\to}m]M\langle\!(\!|m{\otimes}v, x|\!)t\rangle &\to_{\mathtt{axm}_2''} & M\langle\!(\!|n{\otimes}v, x|\!)t\rangle
\end{aligned}
$$

But the ambiguity is harmless and the notation is convenient. Note that in $\to_{\mathtt{axm}_1}$, $\to_\otimes$, and $\to_\multimap$ (and $\to_{\mathtt{axm}_2'/\mathtt{axm}_2''}$), the cut acts on a sub-term inside a multiplicative context $M$, that is, out of !. Moreover, it is silently assumed that $M$ does not capture $m$ in $\to_\otimes$ and $\to_\multimap$ (and $\to_{\mathtt{axm}_2'/\mathtt{axm}_2''}$), but it might capture other variables in $t$ and $v$.

In both $\to_\otimes$ and $\to_\multimap$ the rule has to respect split cuts, which is why, for writing the reduct, the sub-terms $s$ and $u$ are split. We suggest to have a look at the proof of subject reduction on page 24, which shows how some of the rewriting steps act on the decorated proof. An example of $\to_\multimap$ step follows:

$$
[\lambda e.(\!|e?m|\!)m{\to}n](\!|n{\otimes}!f, o|\!)o \to_\multimap [!f{\to}e](\!|e?m|\!)[m{\to}o]o.
$$

**Micro-Step Exponential Rules.** There are also four micro-step exponential rules, with again two rules for axioms. Replacement of variables ($\to_{\mathtt{axe}_1}$) and erasure ($\to_\mathtt{w}$) are expressed generically for exponential values $v_{\mathsf{e}}$, interaction with derelictions (in $\to_{\mathtt{axe}_2}$ and $\to_!$) instead requires inspecting $v_{\mathsf{e}}$. Rule $\to_!$ simply turns into a rewriting rule the relevant case in the definition of meta-level substitution. Note that it entangles *interaction with a dereliction* and *duplication*, which is not what proof nets usually do, as mentioned in Sect. 5. It is silently assumed that $C$ does not capture $e$ in $\to_{\mathtt{axe}_2}$ and $\to_!$ but it might capture other variables in $t$.

**Linear Implication and Cuts Creations.** Rule $\to_{\multimap}$ is special in that it is the only rule of the ESC that *creates* cuts. Note indeed that in the root $\mapsto_{\multimap}$ rule:

$$[\lambda y.L\langle v\rangle\to m]M\langle\!(m\oslash v',x)\!\rangle t\rangle \quad \mapsto_{\multimap} \quad M\langle[v'\to y]L\langle[v\to x]t\rangle\rangle$$

the cut $[v'\to y]$ in the reduct is not a cut of the reducing term. One might argue that also $\mapsto_{\otimes}$ and $\mapsto_!$ do create cuts, because in their root rules, recalled here:

$$[(L\langle v\rangle, L'\langle v'\rangle)\to m]M\langle\!(m\,\aleph\,x,y)\!\rangle t\rangle \quad \mapsto_{\otimes} \quad M\langle L\langle[v\to x]L'\langle[v'\to y]t\rangle\rangle\rangle$$

$$[!L\langle v\rangle\to e]C\langle\!(e?x)\!\rangle t\rangle \quad \mapsto_! \quad [!L\langle v\rangle\to e]C\langle L\langle[v\to x]t\rangle\rangle$$

the cuts $[v\to x]$ and $[v'\to y]$ of the reducts are not in the reducing terms. The difference is that $v$ and $v'$ *do occur as sub-terms of cuts* in the reducing terms of $\mapsto_{\otimes}$ and $\mapsto_!$, while in $\mapsto_{\multimap}$ the value $v'$ does not. We then say that $[v'\to y]$ is *created* in $\mapsto_{\multimap}$. This fact shall play a key role for *local termination* (Theorem 8.8 below) and for the good strategy. Being specific to $\to_{\multimap}$, it is a feature of the intuitionistic setting, that is, it is *invisible* in classical linear logic.

**Positions and Redexes.** For later defining the good strategy, we identify a redex with its position, which is a context, as it is done for the LSC by Accattoli and Dal Lago [ADL16]. Every step $t \to_{\mathtt{ms}_{\neg\mathtt{w}}} s$ reduces a redex of shape $t = C\langle[v\to x]C'\langle t_x\rangle\rangle$ where $t_x$ is an *occurrence* of $x$, i.e. a sub-term of $t$ of shape $x$, $(\!x\,\aleph\,y, z)\!\rangle s$, $(\!x\oslash v, y)\!\rangle s$, or $(\!x?y)\!\rangle s$. The *position* of the $\to_{\mathtt{ms}_{\neg\mathtt{w}}}$ redex is the context $C\langle[v_\mathtt{e}\to x]C'\rangle$. For $\to_\mathtt{w}$ and $\to_{\mathtt{e}^{\mathtt{ss}}}$ redexes, the position is the context closing the root step (as a sub-term). We write $C : t \to_{\mathtt{ms/ss}} s$ for a redex of position $C$.

We conclude the section with the fact that proper terms are stable by reduction.

**Lemma 6.7.** *Let $C\langle t\rangle$ be a proper term. Then:*

(1) *$t$ is proper.*
(2) *If $s$ is proper and such that $\mathtt{mfv}(s) = \mathtt{mfv}(t)$ then $C\langle s\rangle$ is proper.*

*Proof.* By induction on $C$. $\qquad\square$

**Lemma 6.8.** *Let $t$ be a proper term and $t \to_{\mathtt{ms}} s$. Then $s$ is proper.*

*Proof.* Let $t = C\langle u\rangle \to_{\mathtt{ms}} C\langle r\rangle = s$. By Lemma 6.7, $u$ is proper, and to prove the statement it is enough to prove that if $u \to_{\mathtt{ms}} r$ is a root step then $r$ is proper and $\mathtt{mfv}(r) = \mathtt{mfv}(u)$. Note that all the root rules preserve the set of multiplicative variables, that is, $\mathtt{mfv}(r) = \mathtt{mfv}(u)$ holds. Proving that the root rules preserve being proper is a straightforward induction. $\quad\square$

Preservation of properness by $\to_{\mathtt{e}^{\mathtt{ss}}}$ follows from *full composition* (Prop. 7.1), just below.

## 7. Basic Properties

We now prove various basic properties of the ESC. An expected property that we do not prove is the simulation of the LSC by the ESC. It can be obtained via Girard's (call-by-name) encoding of intuitionistic logic into linear logic. It is not surprising and yet it is technical, for reasons (the encoding of applications via subtractions) that are intrinsic to the translation and orthogonal to the features of the ESC, which is why it is omitted.

**Full Composition.** First of all, micro-step exponential rules simulate the small-step one. The ES literature calls this fact *full composition*.

**Proposition 7.1** (Full composition, or micro-step simulates small-step)**.** *If $t \to_{\mathsf{e^{ss}}} s$ then $t \to_{\mathsf{e^{ms}}}^+ s$.*

*Proof.* Let $t = C\langle[v_{\mathsf{e}}{\to}e]u\rangle \to_{\mathsf{e^{ss}}} C\langle\{v_{\mathsf{e}}{\to}e\}u\rangle = s$. We treat the case $v_{\mathsf{e}} = \,!r$, the case $v_{\mathsf{e}} = f$ is analogous. The proof is by induction on $k := |u|_{\mathsf{e}}$. If $k = 0$ then $C\langle[v_{\mathsf{e}}{\to}e]u\rangle \to_{\mathsf{w}} C\langle u\rangle = C\langle\{v_{\mathsf{e}}{\to}e\}u\rangle$. Otherwise, $u$ writes has $C'\langle e\rangle$ or $C'\langle\!(e?x)\!u'\rangle$. Consider the first case. Then:

$$C\langle[!u'{\to}e]C'\langle e\rangle\rangle \to_{\mathtt{axe_1}} C\langle[!u'{\to}e]C'\langle!u'\rangle\rangle =: s'$$

Now note that $s' \to_{\mathsf{e^{ss}}} s$ and that in $s'$ we have $|C\langle!u'\rangle|_{\mathsf{e}} < k$. Then we can apply the *i.h.*, obtaining $s' \to_{\mathsf{e^{ms}}}^+ s$, that is, $t \to_{\mathsf{e^{ms}}}^+ s$. The case $u = C'\langle\!(e?x)\!u'\rangle$ is analogous, simply using $\to_!$ instead of $\to_{\mathtt{axe_1}}$.  □

**Corollary 7.2.** *Let $t$ be a proper term and $t \to_{\mathsf{ss}} s$. Then $s$ is proper.*

**Subject Reduction.** The next property is that the defined untyped cut elimination respects the typing system, that is, subject reduction holds. We first need a lemma.

**Lemma 7.3.** *Let $\pi \rhd \Gamma \vdash t : A$. We have $A = \,!B$ if and only if $t = L\langle v_{\mathsf{e}}\rangle$.*

*Proof.* If $A = \,!B$, then consider $t = L\langle v\rangle$. The proof is by induction on $L$. Clearly, if $L = \langle\cdot\rangle$ then the last rule of $\pi$ is necessarily an exponential axiom or a right introduction of !, that is, $v = e$ or $v = \,!s$, that is, $v$ is an exponential value. If $L \neq \langle\cdot\rangle$ then it follows from the *i.h.*
 The reasoning is analogous if $A \neq \,!B$.  □

**Theorem 7.4** (Subject reduction)**.** *Let $\Gamma \vdash t : A$ and $t \to_{\mathsf{ms}} s$. Then $\Gamma \vdash s : A$.*

*Proof.* By induction on $\pi \rhd \Gamma \vdash t : A$. If $\pi$ is an axiom then $t$ cannot reduce. If the last rule is a cut then $t = L\langle[v{\to}x]s\rangle$. If it is the root cut that it is reduced there are a number of cases given next. Otherwise (that is, if the last rule is not a cut, or if the reduced cut is not the one of the last rule) the statement follows by the *i.h.*
 By Lemma 7.3, if the cut formula is $!B$ for some $B$ then $x = e$ and $v = v_{\mathsf{e}}$, and otherwise $x = m$ and $v = v_{\mathsf{m}}$. We show two representative cases, the $\to_{\multimap}$ one, which shows how to interpret the splitting of terms into value and left contexts on proofs, and $\to_{\mathtt{axe_2}}$, which shows how LSC-style duplication acts on proofs. The other cases are similar.

- $\mapsto_{\multimap}$.

$$\cfrac{\cfrac{\cfrac{\cfrac{\overline{\overline{\Gamma \vdash v : B}}}{\Gamma', y : A \vdash L\langle v\rangle : B}\, L}{\Gamma' \vdash \lambda y.L\langle v\rangle : A \multimap B}\, \multimap_r}{\overline{\overline{\Gamma'' \vdash L'\langle\lambda y.L\langle v\rangle\rangle : A \multimap B}}\, L'} \qquad \cfrac{\cfrac{\cfrac{\overline{\overline{\Delta \vdash v' : A}}}{\Delta' \vdash L''\langle v'\rangle : A}\, L'' \qquad \Pi, x : B \vdash t : C'}{\Delta', \Pi, m : A \multimap B \vdash L''\langle\!(m{\otimes}v', x, t)\!\rangle : C'}\, \multimap_l}{\overline{\overline{\Pi', m : A \multimap B \vdash M\langle L''\langle\!(m{\otimes}v', x, t)\!\rangle\rangle : C'}}\, M}}{\Gamma'', \Pi' \vdash L'\langle[\lambda y.L\langle v\rangle{\to}m]M\langle L''\langle\!(m{\otimes}v', x, t)\!\rangle\rangle\rangle : C'}\, \mathtt{cut}$$

$$\to_{\multimap}$$

$$\cfrac{\cfrac{\cfrac{\overline{\overline{\Delta \vdash v' : A}}}{\Delta' \vdash L''\langle v'\rangle : A}\, L'' \qquad \cfrac{\cfrac{\overline{\overline{\Gamma \vdash v : B}}}{\Gamma', y : A \vdash L\langle v\rangle : B}\, L \qquad \Pi, y : A, x : B \vdash t : C}{\Delta', y : A, \Pi \vdash L\langle[v{\to}x]t\rangle : C'}\, \mathtt{cut}}{\cfrac{\cfrac{\Gamma', \Delta', \Pi \vdash L''\langle[v'{\to}y]L\langle[v{\to}x]t\rangle\rangle}{\overline{\overline{\Gamma', \Pi' \vdash M\langle L''\langle[v'{\to}y]L\langle[v{\to}x]t\rangle\rangle\rangle : C'}}\, M}}{\overline{\overline{\Gamma'', \Pi' \vdash L'\langle M\langle L''\langle[v'{\to}y]L\langle[v{\to}x]t\rangle\rangle\rangle\rangle : C'}}\, L'}}{}\, \mathtt{cut}}$$

- $\to_{\mathtt{axe}_2}$. The step is $L\langle[f{\to}e]C\langle\!(e?x\!)t\rangle\rangle : B \to_{\mathtt{axe}_2} L\langle[f{\to}e]C\langle\!(e?x\!)t\rangle\rangle : B$. To avoid analyzing many similar (simple) sub-cases, let us assume that $L$ does not capture $f$ and that $f \notin \mathtt{fv}(L)$. Now there still are two cases, depending on the multiplicity of $e$ in $C\langle\!(e?x\!)t\rangle$.
  - $e \notin \mathtt{fv}(C)$. Then:

$$\cfrac{\cfrac{\overline{f:!A \vdash f:!A}\ \mathtt{ax_e}}{\Gamma, f:!A \vdash L\langle f\rangle : A}\ L \quad \cfrac{\cfrac{\cfrac{\Pi, x:A \vdash t:B}{\Pi, e:!A \vdash \langle\!(e?x\!)t : B}\ !_l}{\Delta, e:!A \vdash C\langle\!(e?x\!)t\rangle : B}\ C}{\Gamma, f:!A, \Delta \vdash L\langle[f{\to}e]C\langle\!(e?x\!)t\rangle\rangle : B}}{}\ \mathtt{cut}$$

$$\to_{\mathtt{axe}_2}$$

$$\cfrac{\cfrac{\cfrac{\overline{f:!A \vdash f:!A}\ \mathtt{ax_e}}{\Gamma, f:!A \vdash L\langle f\rangle : A}\ L \quad \cfrac{\cfrac{\cfrac{\Pi, x:A \vdash t:B}{\Pi, g:!A \vdash \langle\!(g?x\!)t : B}\ !_l}{\Delta, g:!A \vdash C\langle\!(g?x\!)t\rangle : B}\ C}{\Delta, g:!A, e:!A \vdash C\langle\!(g?x\!)t\rangle : B}\ \mathtt{w}}{\Gamma, f:!A, g:!A, \Delta \vdash L\langle[f{\to}e]C\langle\!(g?x\!)t\rangle\rangle : B}\ \mathtt{cut}}{\Gamma, f:!A, \Delta \vdash L\langle[f{\to}e]C\langle\!(f?x\!)t\rangle\rangle : B}\ \mathtt{c}$$

  - $e \in \mathtt{fv}(C)$. Then in the sequence of rules corresponding to the context $C$ there is a first contraction acting on the dereliction which splits the sequence of rules in two, that is, there exists $C'$ and $C''$ such that $C = C'\langle C''\rangle$ and:

$$\cfrac{\cfrac{\overline{f:!A \vdash f:!A}\ \mathtt{ax_e}}{\Gamma, f:!A \vdash L\langle f\rangle : A}\ L \quad \cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\Pi, x:A \vdash t:B}{\Pi, g:!A \vdash \langle\!(g?x\!)t : B}\ !_l}{\Delta', e':!A, g:!A \vdash C''\langle\!(g?x\!)t\rangle : B}\ C''}{\Delta', e':!A \vdash C''\langle\!(e'?x\!)t\rangle : B}\ \mathtt{c}}{\Delta, e:!A \vdash C'\langle C''\langle\!(e?x\!)t\rangle\rangle : B}\ C'}{}}{\Gamma, f:!A, \Delta \vdash L\langle[f{\to}e]C\langle\!(e?x\!)t\rangle\rangle : B}\ \mathtt{cut}$$

$$\to_{\mathtt{axe}_2}$$

$$\cfrac{\cfrac{\cfrac{\overline{f:!A \vdash f:!A}\ \mathtt{ax_e}}{\Gamma, f:!A \vdash L\langle f\rangle : A}\ L \quad \cfrac{\cfrac{\cfrac{\cfrac{\Pi, x:A \vdash t:B}{\Pi, g:!A \vdash \langle\!(g?x\!)t : B}\ !_l}{\Delta', e':!A, g:!A \vdash C''\langle\!(g?x\!)t\rangle : B}\ C''}{\Delta, e:!A, g:!A \vdash C'\langle C''\langle\!(g?x\!)t\rangle\rangle : B}\ C'}{\Gamma, f:!A, g:!A, \Delta \vdash L\langle[f{\to}e]C\langle\!(g?x\!)t\rangle\rangle : B}\ \mathtt{cut}}{\Gamma, f:!A, \Delta \vdash L\langle[f{\to}e]C\langle\!(f?x\!)t\rangle\rangle : B}}{}\ \mathtt{c}$$

$\square$

Full composition extends subject reduction to $\to_{\mathtt{e^{ss}}}$.

**Clashes.** The presence of many constructors in an untyped setting gives rise to *clashes*, that is, irreducible cuts.

**Definition 7.5** (Clash, clash-free terms). A *clash* is a term of the form $[v_{\mathtt{m}}{\to}e]t$, $[v_{\mathtt{e}}{\to}m]t$, $[(t,s){\to}m]M\langle\!(m\otimes v, y\!)u\rangle$, or $[\lambda x.t{\to}m]M\langle\!(m\,\mathscr{P}\,y, z\!)u\rangle$ and in these cases we also say that the root cut is *clashing*. A term $t$ is *clash-free* if, co-inductively,

(1) No sub-terms of $t$ are clashes, and
(2) If $t \to_{\mathtt{ms}} s$ then $s$ is clash-free.

Note that there are no purely exponential clashes. The second point of the next lemma uses subject reduction.

**Lemma 7.6.** *Let $t$ be a proper term.*

(1) *If $t$ has no clashes and $t \not\to_{\mathtt{ms}}$ then $t$ is cut-free.*
(2) *If $t$ is typable then it is clash-free.*

*Proof.*

(1) Suppose by contradiction that $t$ has a cut. Then $t = C\langle[v{\to}x]s\rangle$. If $x$ is exponential then, since there are no clashes, $v$ is also exponential. Then there is a $\to_{\mathtt{e}^{\mathtt{ss}}}$ redex, and, by full composition, a $\to_{\mathtt{e}^{\mathtt{ms}}}$ redex, absurd. If $x = m$ is multiplicative then so is $v$. By structural linearity (Lemma 6.3), $s$ has one of the following shapes:
   - $M\langle m \rangle$. Then there is a $\to_{\mathtt{axm}_1}$ redex, absurd.
   - $M\langle\!\langle m \,\mathfrak{N}y, z\rangle\!\rangle s\rangle$. Since $t$ has no clashes, $v$ is either a multiplicative variable, and then there is a $\to_{\mathtt{axm}_2}$ redex, or a tensor pair, and then there is a $\to_{\otimes}$ redex, absurd.
   - $M\langle\!\langle m \ominus v, y\rangle\!\rangle s\rangle$. Since $t$ has no clashes, $v$ is either a multiplicative variable, and then there is a $\to_{\mathtt{axm}_2}$ redex, or an abstraction, and then there is a $\to_{\multimap}$ redex, absurd.
(2) By co-induction on the definition of being clash-free. It is easily seen that clashes are not typable. So if $t$ is typable then it has no clashes. Now, if $t \to_{\mathtt{ms}} s$ then by subject reduction (Theorem 7.4) $s$ is typable. By the co-inductive hypothesis, $s$ is clash-free. Therefore, $t$ is clash-free. $\qquad\square$

**Postponement of Garbage Collection.** The micro-step rule $\to_{\mathtt{w}}$, that models garbage collection, can be postponed. The same happens in the LSC, but *not* in the $\lambda$-calculus: for instance, the first step in the sequence $(\lambda x.\lambda y.y)su \to_\beta (\lambda y.y)u \to_\beta u$ is erasing but cannot be postponed.

**Lemma 7.7** (Local postponement of garbage collection). *If $t \to_{\mathtt{w}}\to_{\mathtt{ms}_{\neg\mathtt{w}}} s$ then $t \to_{\mathtt{ms}_{\neg\mathtt{w}}}\to_{\mathtt{w}}^+ s$.*

*Proof.* By induction on $t \to_{\mathtt{w}} u$ and case analysis of $u \to_{\mathtt{ms}_{\neg\mathtt{w}}} s$. Cases:

- *Root*: $t = [v_{\mathtt{e}}{\to}e]u \to_{\mathtt{w}} u \to_{\mathtt{ms}_{\neg\mathtt{w}}} s$. Then $[v_{\mathtt{e}}{\to}e]u \to_{\mathtt{ms}_{\neg\mathtt{w}}} [v_{\mathtt{e}}{\to}e]s \to_{\mathtt{w}} s$.
- *Inductive cases different from cut*: we treat the case of tensor, the others are similar. Let $t = (r, p) \to_{\mathtt{w}} (r', p) = u$. If the $\to_{\mathtt{ms}_{\neg\mathtt{w}}}$ step takes place in $r'$ then we apply the *i.h.*, otherwise $(r', p) \to_{\mathtt{ms}_{\neg\mathtt{w}}} (r', p') = s$ and we simply have $(r, p) \to_{\mathtt{ms}_{\neg\mathtt{w}}} (r, p') \to_{\mathtt{w}} (r', p')$.
- *Inductive case left of cut*: $t = [v{\to}x]p \to_{\mathtt{w}} [v'{\to}x]p = s$. If the $\to_{\mathtt{ms}_{\neg\mathtt{w}}}$ step takes place in $v'$ we use the *i.h.*, and if it is in $p$ then the two steps are disjoint and swap. The last possibility is that the root cut itself is the active one in the $\to_{\mathtt{ms}_{\neg\mathtt{w}}}$ redex. Now, the key observation is that swapping the step $s \to_{\mathtt{ms}_{\neg\mathtt{w}}} u$ before the $\to_{\mathtt{w}}$ one amounts to add a cut (the erased one) to both $u$ and $s$, and to remove it with the postponed $\to_{\mathtt{w}}$ step, which can be done smoothly because the rewriting rules do not depend on the cuts surrounding the sub-terms of the redex nor those surrounding the redex. The only cases where something interesting happens are those for $\to_{\mathtt{axe}_1}$ and $\to_!$ when the $\to_{\mathtt{w}}$ step is in the duplicated exponential value. For instance, if $f \notin \mathtt{fv}(t)$ we have the following diagram.

$$
\begin{array}{ccc}
[V\langle[v_{\mathtt{e}}{\to}f]t\rangle{\to}e]C\langle\!\langle e\rangle\!\rangle & \to_{\mathtt{w}} & [V\langle t\rangle{\to}e]C\langle\!\langle e\rangle\!\rangle \\
\downarrow{\mathtt{axe}_1} & & \downarrow{\mathtt{axe}_1} \\
[V\langle[v_{\mathtt{e}}{\to}f]t\rangle{\to}e]C\langle\!\langle V\langle[v_{\mathtt{e}}{\to}f]t\rangle\rangle\!\rangle \dashrightarrow_{\mathtt{w}} [V\langle t\rangle{\to}e]C\langle\!\langle V\langle[v_{\mathtt{e}}{\to}f]t\rangle\rangle\!\rangle \dashrightarrow_{\mathtt{w}} [V\langle t\rangle{\to}e]C\langle\!\langle V\langle t\rangle\rangle\!\rangle
\end{array}
$$

- *Inductive case right of cut*: $t = [v{\to}x]p \to_{\mathtt{w}} [v{\to}x]p' = s$. If the $\to_{\mathtt{ms}_{\neg\mathtt{w}}}$ step takes place in $p'$ we use the *i.h.*, and if it is in $v$ then the two steps are disjoint and swap. The last possibility is that the root cut itself is the active one in the $\to_{\mathtt{ms}_{\neg\mathtt{w}}}$ redex. As in the previous case the swap happens smoothly because the rewriting rules do not depend on

the cuts surrounding the subterms of the redex nor those surrounding the redex. There are no interesting cases. □

**Proposition 7.8** (Postponement of garbage collection). *if* $t \to_{\mathtt{ms}}^* s$ *then* $t \to_{\mathtt{ms} \neg \mathtt{w}}^* \to_{\mathtt{w}}^* s$.

*Proof.* It is an instance of a well-known rewriting property: the local swap in Lemma 7.7 is an instance of Hindley's strong postponement property, which implies postponement. □

**Cut Equivalence.** Similarly to the structural equivalence of the LSC, we consider a notion of cut equivalence, which is a strong bisimulation and it is postponable.

**Definition 7.9** (Cut equivalence). If $x \notin \mathtt{fv}(M)$ and $M$ does not capture variables in $\mathtt{fv}(v)$, define:

$$[v \to x]M\langle t \rangle \quad \sim_{\mathtt{cut}} \quad M\langle [v \to x]t \rangle$$

*Cut equivalence* $\equiv_{\mathtt{cut}}$ is the closure of $\sim_{\mathtt{cut}}$ by reflexivity, symmetry, transitivity, and general contexts.

To prove that cut equivalence is a strong bisimulation in the small-step case, we need the following substitutivity property.

**Lemma 7.10** (Stability by substitution of $\equiv_{\mathtt{cut}}$).

(1) *If* $t \equiv_{\mathtt{cut}} s$ *then* $\{v_{\mathtt{e}} \to e\}t \equiv_{\mathtt{cut}} \{v_{\mathtt{e}} \to e\}s$ *for all* $v_{\mathtt{e}}$.
(2) *If* $v_{\mathtt{e}} \equiv_{\mathtt{cut}} v_{\mathtt{e}}'$ *then* $\{v_{\mathtt{e}} \to e\}t \equiv_{\mathtt{cut}} \{v_{\mathtt{e}}' \to e\}t$.

*Proof.* Easy inductions on $t \equiv_{\mathtt{cut}} s$ and $t$. Details in [Acc22]. □

**Proposition 7.11.** *Let* $\to \in \{\to_{\mathtt{ss}}, \to_{\mathtt{ms}}\}$.

(1) Strong bisimulation*:* $\equiv_{\mathtt{cut}}$ *is a strong bisimulation with respect to* $\to$*, preserving also the kind of step.*
(2) Preservation of SN*: if* $t \equiv_{\mathtt{cut}} s$ *and* $t \in SN_{\to}$ *then* $s \in SN_{\to}$.
(3) Postponement of $\equiv_{\mathtt{cut}}$*: if* $t(\equiv_{\mathtt{cut}} \to \equiv_{\mathtt{cut}})^k s$ *then* $t \to^k \equiv_{\mathtt{cut}} s$.

*Proof.*

(1) It is a very long but otherwise straightforward check of all possible diagrams, spelled out in [Acc22]. The following two diagrams for $\to_{\mathtt{e}^{\mathtt{ss}}}$ are where Lemma 7.10 is used.

$$
\begin{array}{cccc}
[v \to y]s & \to_{\mathtt{e}^{\mathtt{ss}}} & \{v \to y\}s & \qquad\qquad [v \to y]s & \to_{\mathtt{e}^{\mathtt{ss}}} & \{v \to y\}s \\
\equiv_{\mathtt{cut}} & & \equiv_{\mathtt{cut}} & \qquad\qquad \equiv_{\mathtt{cut}} & & \equiv_{\mathtt{cut}} \\
[v \to y]s' & \to_{\mathtt{e}^{\mathtt{ss}}} & \{v \to y\}s' & \qquad\qquad [v' \to y]s & \to_{\mathtt{e}^{\mathtt{ss}}} & \{v' \to y\}s
\end{array}
$$

(2) By induction on $t \in SN_{\to}$. One simply uses the fact that $\equiv_{\mathtt{cut}}$ is a strong bisimulation, given by the previous point, and the *i.h.*
(3) By induction on $k$, using the strong bisimulation property. □

Cut equivalence shall play a role in the following sections. Note that its postponement property means that it is *never needed* for cut elimination to progress. And on cut-free terms, obviously, it vanishes.

There is also a larger *left equivalence* $\equiv_{\mathtt{left}}$ defined as $\equiv_{\mathtt{cut}}$ but including every left constructor (cut, par, subtraction, and dereliction), which is also a strong bisimulation. As mentioned in Sect. 4, $\equiv$ is the quotient induced by proof nets on the LSC. Intuitively, $\equiv_{\mathtt{left}}$ is the quotient induced by proof nets on the ESC. There are, however, no proof nets realizing

such a quotient, as weakenings require non-canonical *jumps* for IMELL, that can instead be avoided for the LSC, as shown by Accattoli [Acc18].

A possibly interesting point is that we shall need cut equivalence, in particular for the proof of strong normalization, while there is no need of left equivalence for any of the properties studied in this paper.

**Stability Under Renaming.** The last basic property that we prove is the stability of the rules and of their strong normalization under renaming.

**Lemma 7.12** (Renaming). *Let* $\to \in \{\to_{\mathtt{ss}}, \to_{\mathtt{ms}}\}$ *and* $x$ *and* $y$ *be two variables of the same multiplicative/exponential kind such that* $\{x{\to}y\}t$ *is proper.*

(1) Stability of steps under renaming: *if* $C : t \to_a s$ *then* $\{y{\to}x\}C : \{y{\to}x\}t \to_a \{y{\to}x\}s$.
(2) Stability of SN under renaming: *if* $t \in SN_{\to_{\mathtt{ss}}}$ *then* $\{x{\to}y\}t \in SN_{\to_{\mathtt{ss}}}$.

*Proof.*

(1) Straightforward induction on the rewriting step.
(2) By induction on $E\langle\!\langle t \rangle\!\rangle \in SN_{\to_{\mathtt{ss}}}$. Since renaming cannot create, erase, duplicate, or change the kind of redexes, if $\{x{\to}y\}t \to_{\mathtt{ss}} s$ then there exists $u$ such that $t \to_{\mathtt{ss}} u$ and $\{x{\to}y\}u = s$. Then by *i.h.* on $u$ we obtain $\{x{\to}y\}u = s \in SN_{\to_{\mathtt{ss}}}$. $\qquad\square$

## 8. Local termination

Here we are going to prove that the ESC has the same local termination property of the LSC, thus showing that one can claim that it realizes the slogan *exponentials as substitutions*.

**Divergence.** Let us first show that evaluation in the untyped ESC can diverge. We adapt the usual looping combinator $\Omega$ of the $\lambda$-calculus. Let $\delta := \lambda e.(\!|e?m|\!)(\!|m \otimes e, n|\!)n$ and consider:

$$
\begin{aligned}
\Omega \quad := \quad & [\delta{\to}o](\!|o\otimes!\delta, o'|\!)o' & \to_{\multimap} \quad & [!\delta{\to}e](\!|e?m|\!)(\!|m\otimes e, n|\!)[n{\to}o']o' \\
& & \to_{\mathtt{axe}_1} \quad & [!\delta{\to}e](\!|e?m|\!)(\!|m\otimes!\delta, n|\!)[n{\to}o']o' \\
& & \to_! \quad & [!\delta{\to}e][\delta{\to}m](\!|m\otimes!\delta, n|\!)[n{\to}o']o' \\
& & \to_{\mathtt{w}} \quad & [\delta{\to}m](\!|m\otimes!\delta, n|\!)[n{\to}o']o' \\
& & \to_{\mathtt{axm}_1} \quad & [\delta{\to}m](\!|m\otimes!\delta, n|\!)n & =_\alpha \quad \Omega
\end{aligned}
$$

**Local Termination.** A key feature of the untyped ESC (inherited from the LSC) is *local termination*, that is, every rewriting rule is strongly normalizing (SN) when considered separately (not valid in the $\lambda$-calculus which has only one rule). Additionally, the groups of multiplicative rules $\to_{\mathtt{m}}$ and of micro-step exponential rules $\to_{\mathtt{e}^{\mathtt{ms}}}$ are also SN separately, which is instead not valid in untyped classical proof nets (Fig. 1 left).

Even the union of *all* the micro-step rules except $\to_{\multimap}$, noted $\to_{\mathtt{ms}_{\neg\multimap}}$, is SN. This is due to the fact that $\to_{\multimap}$ is the only rule that *creates* cuts, as explained in Sect. 6. Strong normalization of $\to_{\mathtt{ms}_{\neg\multimap}}$ is a deep property, pointed out here for the first time. It shows that, for expressivity, rule $\to_{\multimap}$ is as important as duplication, and thus, non-linearity. It also shows that $\multimap$ and $\otimes$ have completely different computational properties. This is remarkable because in classical linear logic *à la* Girard, the two are collapsed and managed by a single cut elimination rule, the one for $\mathfrak{N}/\otimes$, and creations are not visible. Together with (non-)termination of untyped exponentials, it is a sign that the intuitionistic setting is sharper than the classical one for studying cut elimination.

**Proving Local Termination.** The proofs of local termination and of the termination of $\to_{\mathtt{ms}\neg\multimap}$ require various definitions. First, we need a notion of *variable potential*, which is used to define a *measure*, that it is then showed to decrease with every micro-step rewriting rule but $\to_{\multimap}$. A number of auxiliary lemmas are also required.

The intuition behind the potential $\mathtt{p}_x(t)$ of a variable $x$ in $t$ is that it is the maximum number of occurrences of $x$ that can appear during the cut elimination of $t$, plus the same number for all the variables that are recursively 'nested' under $x$, where for instance $x$ nests under $e$ in $(\!| e?x|\!)t$. The key clause is the one for cuts $[v{\to}x]t$, which multiplies the potential in $v$ for the one of $x$ in $t$.

**Definition 8.1** (Variable potential). The potential $\mathtt{p}_x(t)$ of a variable $x$ in $t$ is given by:

$$
\begin{aligned}
\mathtt{p}_x(y) &:= 0 \\
\mathtt{p}_x(x) &:= 1 \\
\mathtt{p}_x((t,s)) &:= \mathtt{p}_x(t) + \mathtt{p}_x(s) \\
\mathtt{p}_x(!t) &:= \mathtt{p}_x(t) \\
\mathtt{p}_x(\lambda y.t) &:= \mathtt{p}_x(t)
\end{aligned}
$$

$$
\mathtt{p}_x((\!|m\gamma y, z|\!)t) := \begin{cases} 1 + \mathtt{p}_y(t) + \mathtt{p}_z(t) & \text{if } x = m \\ \mathtt{p}_x(t) & \text{otherwise.} \end{cases}
$$

$$
\mathtt{p}_x((\!|m\oslash v, y|\!)t) := \begin{cases} 1 & \text{if } x = m \\ \mathtt{p}_x(v) + \mathtt{p}_x(t) & \text{otherwise.} \end{cases}
$$

$$
\mathtt{p}_x((\!|e?y|\!)t) := \begin{cases} 1 + \mathtt{p}_x(t) + \mathtt{p}_y(t) & \text{if } x = e \\ \mathtt{p}_x(t) & \text{otherwise.} \end{cases}
$$

$$
\mathtt{p}_x([v{\to}y]t) := \mathtt{p}_x(t) + \mathtt{p}_x(v) \cdot (\mathtt{p}_y(t) + 1)
$$

The potential is extended to contexts by considering $\langle\cdot\rangle$ as a (fresh) exponential variable, that is, having $\mathtt{p}_x(\langle\cdot\rangle) := 0$, $\mathtt{p}_{\langle\cdot\rangle}(x) := 0$, and $\mathtt{p}_{\langle\cdot\rangle}(\langle\cdot\rangle) := 1$, plus all the expected inductive cases for contexts.

In the definition of the potential, the clause for cut multiplies $\mathtt{p}_x(v)$ for $\mathtt{p}_y(t) + 1$ where, for exponential variables, the $+1$ accounts for the garbage copy of $v$ that survives the replacement of all the occurrences of $x$ with $v$. The clause does not distinguish between cuts on multiplicative or exponential variables, treating all variables as if they were exponential. Such an over-estimate is harmless and provides a more compact definition of the potential.

Note also the clause for subtractions: when $x = m$, one could define it as 0, because anyway cuts on subtractions are ignored. But it simpler to set it to 1, as to obtain the useful property of Lemma 8.3.2 below.

**Definition 8.2** (Termination measure). The termination measure $\mathtt{ms}(t)$ of a term $t$ is defined as follows (and extended to contexts by defining it also for $\langle\cdot\rangle$):

$$
\begin{aligned}
\mathtt{ms}(x) &:= 1 & \mathtt{ms}(\langle\cdot\rangle) &:= 0 \\
\mathtt{ms}((\!|m\gamma x, y|\!)t) &:= \mathtt{ms}(t) + 1 & \mathtt{ms}((t,s)) &:= \mathtt{ms}(t) + \mathtt{ms}(s) \\
\mathtt{ms}((\!|m\oslash v, x|\!)t) &:= \mathtt{ms}(v) + \mathtt{ms}(t) + 1 & \mathtt{ms}(\lambda x.t) &:= \mathtt{ms}(t) \\
\mathtt{ms}((\!|e?x|\!)t) &:= \mathtt{ms}(t) + 1 & \mathtt{ms}(!t) &:= \mathtt{ms}(t) \\
\mathtt{ms}([v{\to}x]t) &:= \mathtt{ms}(v) \cdot (\mathtt{p}_x(t) + 1) + \mathtt{ms}(t)
\end{aligned}
$$

The idea behind the measure is that it counts all variable occurrences (that is, variables, pars, derelictions, and subtractions), multiplying for the potential of $x$ the occurrences appearing in a value cut on $x$, as for the potential. Each rewriting step removes a variable occurrence, which makes the measure decrease. An exponential rewriting step can also duplicate a sub-term and increase the number of occurrences, but this increment is anticipated by the measure by means of the potential.

For proving that the measure decreases, we need some basic properties of the potential and of the measure plus some lemmas about their decomposition with respect to contexts.

**Lemma 8.3** (Basic properties).
(1) *if $x \notin \mathtt{fv}(t)$ then $\mathtt{p}_x(t) = 0$.*
(2) *if $x \in \mathtt{fv}(t)$ then $\mathtt{p}_x(t) \geq 1$.*
(3) $\mathtt{p}_{\langle \cdot \rangle}(C) \geq 1$.
(4) $\mathtt{ms}(t) \geq 1$.
(5) $\mathtt{ms}(t) = \mathtt{ms}(\{n \to m\}t)$.

*Proof.* Points 1, 2, 4, and 5 are easy inductions on $t$. Point 3 follows from Point 2, since, when looking at the hole $\langle \cdot \rangle$ as a variable, one has $\langle \cdot \rangle \in \mathtt{fv}(C)$ for every context $C$. $\qquad \square$

**Lemma 8.4** (Properties for the $\to_{\mathtt{axe}_1}$ and $\to_{\mathtt{axm}_1}$ cases).
(1) $\mathtt{p}_x(C \langle\!\langle t \rangle\!\rangle) = \mathtt{p}_x(C) + \mathtt{p}_{\langle \cdot \rangle}(C) \cdot \mathtt{p}_x(t)$.
(2) $\mathtt{ms}(C \langle\!\langle t \rangle\!\rangle) = \mathtt{ms}(C) + \mathtt{p}_{\langle \cdot \rangle}(C) \cdot \mathtt{ms}(t)$.

*Proof.* Both points are by induction on $C$, and the second one uses the first one in the cut case. Details are in [Acc22]. $\qquad \square$

**Lemma 8.5** (Properties for the $\to_\otimes$, $\to_{\mathtt{axe}_2}$, $\to_!$ cases).
(1) *If $C$ does not capture variables in $\mathtt{fv}(L)$ and $x \notin \mathtt{fv}(L)$ then $\mathtt{p}_x(C \langle L \langle t \rangle \rangle) = \mathtt{p}_x(C \langle t \rangle)$.*
(2) *If $C$ does not capture $e$ and $f$ then:*
    (a) *Potential:* $\mathtt{p}_e(C \langle (\!|e?x|\!)t \rangle) = \mathtt{p}_e(C \langle t \rangle) + (1 + \mathtt{p}_x(t)) \cdot \mathtt{p}_{\langle \cdot \rangle}(C)$ *and* $\mathtt{p}_e(C \langle (\!|f?x|\!)t \rangle) < \mathtt{p}_e(C \langle (\!|e?x|\!)t \rangle)$.
    (b) *Measure:* $\mathtt{ms}(C \langle (\!|e?x|\!)t \rangle) > \mathtt{ms}(C \langle t \rangle)$ *and* $\mathtt{ms}(C \langle (\!|e?x|\!)t \rangle) = \mathtt{ms}(C \langle (\!|f?x|\!)t \rangle)$.
(3) *If $M$ does not capture $m$ then:*
    (a) *Potential:* $\mathtt{p}_m(M \langle (\!|m \mathbin{⅋} x, y|\!)t \rangle) = (1 + \mathtt{p}_x(t) + \mathtt{p}_y(t)) \cdot \mathtt{p}_{\langle \cdot \rangle}(M)$.
    (b) *Measure:* $\mathtt{ms}(M \langle (\!|m \mathbin{⅋} x, y|\!)t \rangle) > \mathtt{ms}(M \langle t \rangle)$.

*Proof.* All points are by induction on the context of the statement. The only non-trivial case, for each point, is the one for cut, which is treated in [Acc22]. The others follow immediately from the *i.h.* $\qquad \square$

    The fourth point of the next lemma shall be used in the proof that the measure decreases for the the $\to_\otimes$ and $\to_!$ cases. The first three point are intermediate properties to obtain the fourth one.

**Lemma 8.6** (Inequalities for the $\to_\otimes$ and $\to_!$ cases). *Let $C$ be a context that does not capture free variables in $L$ nor $v$, and $L$ does not capture variables of $t$.*
(1) $\mathtt{p}_y(L \langle [v \to x]t \rangle) \leq \mathtt{p}_y(t) + \mathtt{p}_y(L \langle v \rangle) \cdot (1 + \mathtt{p}_x(t))$.
(2) $\mathtt{ms}(L \langle [v \to x]t \rangle) \leq \mathtt{ms}(t) + \mathtt{ms}(L \langle v \rangle) \cdot (1 + \mathtt{p}_x(t))$.
(3) $\mathtt{p}_y(C \langle L \langle [v \to x]t \rangle \rangle) \leq \mathtt{p}_y(C \langle t \rangle) + \mathtt{p}_y(L \langle v \rangle) \cdot (1 + \mathtt{p}_x(t)) \cdot \mathtt{p}_{\langle \cdot \rangle}(C)$.
(4) $\mathtt{ms}(C \langle L \langle [v \to x]t \rangle \rangle) \leq \mathtt{ms}(C \langle t \rangle) + \mathtt{ms}(L \langle v \rangle) \cdot (1 + \mathtt{p}_x(t)) \cdot \mathtt{p}_{\langle \cdot \rangle}(C)$.

*Proof.* Point 1 is by induction on $L$, point 2 uses point 1, point 3 and 4 are by induction on $C$ using point 1 and 2 in the base case. Details are in [Acc22]. $\qquad \square$

**Proposition 8.7** (Measure decreases). *If $t \to_a s$ with $a \in \{\mathtt{axm}_1, \mathtt{axm}_2, \otimes, \mathtt{axe}_1, \mathtt{axe}_2, !, \mathtt{w}\}$ then $\mathtt{ms}(t) > \mathtt{ms}(s)$.*

*Proof.* By induction on $t \to_a s$. All the inductive cases follow immediately from the *i.h.* Root cases:

- $\mapsto_{\mathtt{axm}_1}$, that is, $[v_\mathtt{m} \to m]M \langle\!\langle m \rangle\!\rangle \mapsto_{\mathtt{axm}_1} M \langle\!\langle v_\mathtt{m} \rangle\!\rangle$. Then:

$$
\begin{aligned}
\mathtt{ms}([v_\mathtt{m}{\to}m]M\langle\!\langle m\rangle\!\rangle) = \; & \mathtt{ms}(v_\mathtt{m}) \cdot (\mathtt{p}_m(M\langle\!\langle m\rangle\!\rangle) + 1) + \mathtt{ms}(M\langle\!\langle m\rangle\!\rangle) && =_{L.8.4.1} \\
& \mathtt{ms}(v_\mathtt{m}) \cdot (\; \underbrace{\mathtt{p}_m(M)}_{=_{L.8.3.1}0} + \mathtt{p}_{\langle\cdot\rangle}(M) \cdot \underbrace{\mathtt{p}_m(m)}_{=1} + 1) + \mathtt{ms}(M\langle\!\langle m\rangle\!\rangle) && = \\
& \mathtt{ms}(v_\mathtt{m}) \cdot (\mathtt{p}_{\langle\cdot\rangle}(M) + 1) + \mathtt{ms}(M\langle\!\langle m\rangle\!\rangle) && =_{L.8.4.2} \\
& \underbrace{\mathtt{ms}(v_\mathtt{m})}_{=_{L.8.3.4}1} \cdot (\mathtt{p}_{\langle\cdot\rangle}(M) + 1) + \mathtt{ms}(M) + \underbrace{\mathtt{p}_{\langle\cdot\rangle}(M)}_{\geq_{L.8.3.3}1} \cdot \mathtt{ms}(m) && > \\
& \mathtt{ms}(v_\mathtt{m}) \cdot \mathtt{p}_{\langle\cdot\rangle}(M) + \mathtt{ms}(M) && =_{L.8.4.2} \\
& \mathtt{ms}(M\langle\!\langle v_\mathtt{m}\rangle\!\rangle)
\end{aligned}
$$

- $\mapsto_{\mathtt{axm}_2}$, that is, $[n{\to}m]u \mapsto_{\mathtt{axm}_2} \{n{\to}m\}u$. Then:

$$
\begin{aligned}
\mathtt{ms}([n{\to}m]u) \;=\; & \underbrace{\mathtt{ms}(n)}_{=1} \cdot (\mathtt{p}_m(u) + 1) + \mathtt{ms}(u) \;= \\
& \mathtt{p}_m(u) + 1 + \mathtt{ms}(u) \qquad\qquad > \; \mathtt{ms}(u) \;=_{L.8.3.5} \; \mathtt{ms}(\{n{\to}m\}u)
\end{aligned}
$$

- $\mapsto_{\otimes}$, that is, $[(u,r){\to}m]M\langle(\!|m\,\mathfrak{V}x,y|\!)p\rangle \mapsto_{\otimes} M\langle L\langle[v{\to}x]L'\langle[v'{\to}y]p\rangle\rangle\rangle$ with $u = L\langle v\rangle$ and $r = L'\langle v'\rangle$. Then:

$$
\begin{aligned}
& \mathtt{ms}([(u,r){\to}m]M\langle(\!|m\,\mathfrak{V}x,y|\!)p\rangle) && = \\
& \mathtt{ms}(M\langle(\!|m\,\mathfrak{V}x,y|\!)p\rangle) + \mathtt{ms}((u,r)) \cdot (\mathtt{p}_m(M\langle(\!|m\,\mathfrak{V}x,y|\!)p\rangle) + 1) && >_{L.8.5.3b} \\
& \mathtt{ms}(M\langle p\rangle) + \mathtt{ms}((u,r)) \cdot (\mathtt{p}_m(M\langle(\!|m\,\mathfrak{V}x,y|\!)p\rangle) + 1) && = \\
& \mathtt{ms}(M\langle p\rangle) + (\mathtt{ms}(u) + \mathtt{ms}(r)) \cdot (\mathtt{p}_m(M\langle(\!|m\,\mathfrak{V}x,y|\!)p\rangle) + 1) && =_{L.8.5.3a} \\
& \mathtt{ms}(M\langle p\rangle) + \underbrace{(\mathtt{ms}(u) + \mathtt{ms}(r))}_{\geq_{L.8.3.4}1} \cdot ((1 + \mathtt{p}_x(p) + \mathtt{p}_y(p)) \cdot \mathtt{p}_{\langle\cdot\rangle}(M) + 1) && > \\
& \mathtt{ms}(M\langle p\rangle) + (\mathtt{ms}(u) + \mathtt{ms}(r)) \cdot (1 + \mathtt{p}_x(p) + \mathtt{p}_y(p)) \cdot \mathtt{p}_{\langle\cdot\rangle}(M) && = \\
& \mathtt{ms}(M\langle p\rangle) + \mathtt{ms}(u) \cdot (1 + \mathtt{p}_x(p) + \mathtt{p}_y(p)) \cdot \mathtt{p}_{\langle\cdot\rangle}(M) + \mathtt{ms}(r) \cdot (1 + \mathtt{p}_x(p) + \mathtt{p}_y(p)) \cdot \mathtt{p}_{\langle\cdot\rangle}(M) && \geq \\
& \mathtt{ms}(M\langle p\rangle) + \mathtt{ms}(u) \cdot (1 + \mathtt{p}_y(p)) \cdot \mathtt{p}_{\langle\cdot\rangle}(M) + \mathtt{ms}(r) \cdot (1 + \mathtt{p}_x(p)) \cdot \mathtt{p}_{\langle\cdot\rangle}(M) && = \\
& \underbrace{\mathtt{ms}(M\langle p\rangle) + \mathtt{ms}(L'\langle v'\rangle) \cdot (1 + \mathtt{p}_y(p)) \cdot \mathtt{p}_{\langle\cdot\rangle}(M)}_{=_{L.8.6.4}\mathtt{ms}(M\langle L'\langle[v'{\to}y]p\rangle\rangle)} + \mathtt{ms}(L\langle v\rangle) \cdot (1 + \mathtt{p}_x(p)) \cdot \mathtt{p}_{\langle\cdot\rangle}(M) && = \\
& \mathtt{ms}(M\langle L'\langle[v'{\to}y]p\rangle\rangle) + \mathtt{ms}(L\langle v\rangle) \cdot (1 + \mathtt{p}_x(p)) \cdot \mathtt{p}_{\langle\cdot\rangle}(M) && =_{L.8.5.1} \\
& \mathtt{ms}(M\langle L'\langle[v'{\to}y]p\rangle\rangle) + \mathtt{ms}(L\langle v\rangle) \cdot (1 + \mathtt{p}_x(L'\langle[v'{\to}y]p\rangle)) \cdot \mathtt{p}_{\langle\cdot\rangle}(M) && =_{L.8.6.4} \\
& \mathtt{ms}(M\langle L\langle[v{\to}x]L'\langle[v'{\to}y]p\rangle\rangle\rangle)
\end{aligned}
$$

- $\mapsto_{\mathtt{axe}_1}$, that is, $[v_\mathtt{e}{\to}e]C\langle\!\langle e\rangle\!\rangle \mapsto_{\mathtt{axe}_1} [v_\mathtt{e}{\to}e]C\langle\!\langle v_\mathtt{e}\rangle\!\rangle$.

$$\mathtt{ms}([v_\mathsf{e}{\to}e]C\langle\!\langle e\rangle\!\rangle) \qquad\qquad\qquad\qquad =$$
$$\mathtt{ms}(v_\mathsf{e})\cdot(\mathtt{p}_e(C\langle\!\langle e\rangle\!\rangle)+1)+\mathtt{ms}(C\langle\!\langle e\rangle\!\rangle) \qquad =_{L.8.4.1}$$
$$\mathtt{ms}(v_\mathsf{e})\cdot(\mathtt{p}_e(C)+\mathtt{p}_{\langle\cdot\rangle}(C)\cdot\underbrace{\mathtt{p}_e(e)}_{=1}+1)+\mathtt{ms}(C\langle\!\langle e\rangle\!\rangle) \qquad =$$
$$\mathtt{ms}(v_\mathsf{e})\cdot(\mathtt{p}_e(C)+\mathtt{p}_{\langle\cdot\rangle}(C)+1)+\mathtt{ms}(C\langle\!\langle e\rangle\!\rangle) \qquad =$$
$$\mathtt{ms}(v_\mathsf{e})\cdot(\mathtt{p}_e(C)+1)+\mathtt{ms}(v_\mathsf{e})\cdot\mathtt{p}_{\langle\cdot\rangle}(C)+\mathtt{ms}(C\langle\!\langle e\rangle\!\rangle) \qquad =_{L.8.4.2}$$
$$\mathtt{ms}(v_\mathsf{e})\cdot(\mathtt{p}_e(C)+1)+\mathtt{ms}(v_\mathsf{e})\cdot\mathtt{p}_{\langle\cdot\rangle}(C)+\mathtt{ms}(C)+\mathtt{p}_{\langle\cdot\rangle}(C)\cdot\underbrace{\mathtt{ms}(e)}_{=1} \qquad =$$
$$\mathtt{ms}(v_\mathsf{e})\cdot(\mathtt{p}_e(C)+1)+\underbrace{\mathtt{ms}(v_\mathsf{e})\cdot\mathtt{p}_{\langle\cdot\rangle}(C)+\mathtt{ms}(C)}_{=_{L.8.4.2}\mathtt{ms}(C\langle v_\mathsf{e}\rangle)}+\mathtt{p}_{\langle\cdot\rangle}(C) \qquad =$$
$$\mathtt{ms}(v_\mathsf{e})\cdot(\mathtt{p}_e(C)+1)+\mathtt{ms}(C\langle v_\mathsf{e}\rangle)+\underbrace{\mathtt{p}_{\langle\cdot\rangle}(C)}_{>_{L.8.3.3}0} \qquad >$$
$$\mathtt{ms}(v_\mathsf{e})\cdot(\mathtt{p}_e(C)+1)+\mathtt{ms}(C\langle v_\mathsf{e}\rangle) \qquad =_{L.8.3.1}$$
$$\mathtt{ms}(v_\mathsf{e})\cdot(\mathtt{p}_e(C)+\mathtt{p}_{\langle\cdot\rangle}(C)\cdot\underbrace{\mathtt{p}_e(v_\mathsf{e})}_{=_{L.8.3.1}0}+1)+\mathtt{ms}(C\langle\!\langle v_\mathsf{e}\rangle\!\rangle) \qquad =_{L.8.4.1}$$
$$\mathtt{ms}(v_\mathsf{e})\cdot(\mathtt{p}_e(C\langle\!\langle v_\mathsf{e}\rangle\!\rangle)+1)+\mathtt{ms}(C\langle\!\langle v_\mathsf{e}\rangle\!\rangle) \qquad =$$
$$\mathtt{ms}([v_\mathsf{e}{\to}e]C\langle\!\langle v_\mathsf{e}\rangle\!\rangle)$$

- $\mapsto_{\mathtt{axe_2}}$, that is, $[f{\to}e]C\langle(\!|e?x|\!)t\rangle \mapsto_{\mathtt{axe_2}} [f{\to}e]C\langle(\!|f?x|\!)t\rangle$. Then:

$$\mathtt{ms}([f{\to}e]C\langle(\!|e?x|\!)t\rangle) \;=\; \mathtt{ms}(f)\cdot(\mathtt{p}_e(C\langle(\!|e?x|\!)t\rangle)+1)+\mathtt{ms}(C\langle(\!|e?x|\!)t\rangle) \quad >_{L.8.5.2a}$$
$$\mathtt{ms}(f)\cdot(\mathtt{p}_e(C\langle(\!|f?x|\!)t\rangle)+1)+\mathtt{ms}(C\langle(\!|e?x|\!)t\rangle) \quad =_{L.8.5.2b}$$
$$\mathtt{ms}(f)\cdot(\mathtt{p}_e(C\langle(\!|f?x|\!)t\rangle)+1)+\mathtt{ms}(C\langle(\!|f?x|\!)t\rangle) \quad =$$
$$\mathtt{ms}([f{\to}e]C\langle(\!|f?x|\!)t\rangle)$$

- $\mapsto_!$, that is, $[!s{\to}e]C\langle(\!|e?x|\!)t\rangle \mapsto_! [!s{\to}e]C\langle L\langle[v{\to}x]t\rangle\rangle$ with $s=L\langle v\rangle$. Then:

$$\mathtt{ms}([!s{\to}e]C\langle(\!|e?x|\!)t\rangle) \qquad =$$
$$\mathtt{ms}(!s)\cdot(\mathtt{p}_e(C\langle(\!|e?x|\!)t\rangle)+1)+\mathtt{ms}(C\langle(\!|e?x|\!)t\rangle) \qquad >_{L.8.5.2b}$$
$$\mathtt{ms}(!s)\cdot(\mathtt{p}_e(C\langle(\!|e?x|\!)t\rangle)+1)+\mathtt{ms}(C\langle t\rangle) \qquad =_{L.8.5.2a}$$
$$\mathtt{ms}(!s)\cdot(\mathtt{p}_e(C\langle t\rangle)+(1+\mathtt{p}_x(t))\cdot\mathtt{p}_{\langle\cdot\rangle}(C)+1)+\mathtt{ms}(C\langle t\rangle) \qquad =$$
$$\mathtt{ms}(!s)\cdot(\mathtt{p}_e(C\langle t\rangle)+1)+\mathtt{ms}(!s)\cdot(1+\mathtt{p}_x(t))\cdot\mathtt{p}_{\langle\cdot\rangle}(C)+\mathtt{ms}(C\langle t\rangle) \qquad =$$
$$\mathtt{ms}(!s)\cdot(\mathtt{p}_e(C\langle t\rangle)+1)+\mathtt{ms}(s)\cdot(1+\mathtt{p}_x(t))\cdot\mathtt{p}_{\langle\cdot\rangle}(C)+\mathtt{ms}(C\langle t\rangle) \qquad =$$
$$\mathtt{ms}(!s)\cdot(\mathtt{p}_e(C\langle t\rangle)+1)+\underbrace{\mathtt{ms}(L\langle v\rangle)\cdot(1+\mathtt{p}_x(t))\cdot\mathtt{p}_{\langle\cdot\rangle}(C)+\mathtt{ms}(C\langle t\rangle)}_{=_{L.8.6.4}\mathtt{ms}(C\langle L\langle[v{\to}x]t\rangle\rangle)} \qquad =$$
$$\mathtt{ms}(!s)\cdot(\mathtt{p}_e(C\langle t\rangle)+1)+\mathtt{ms}(C\langle L\langle[v{\to}x]t\rangle\rangle) \qquad =_{L.8.5.1}$$
$$\mathtt{ms}(!s)\cdot(\mathtt{p}_e(C\langle L\langle[v{\to}x]t\rangle\rangle)+1)+\mathtt{ms}(C\langle L\langle[v{\to}x]t\rangle\rangle) \qquad =$$
$$\mathtt{ms}([!s{\to}e]C\langle L\langle[v{\to}x]t\rangle\rangle)$$

- $\mapsto_\mathtt{w}$, that is, $[v_\mathsf{e}{\to}e]t \mapsto_\mathtt{w} t$ with $e\notin\mathtt{fv}(t)$. Then:

$$\mathtt{ms}([v_\mathsf{e}{\to}e]t) \;=\; \mathtt{ms}(v_\mathsf{e})\cdot(\mathtt{p}_e(t)+1)+\mathtt{ms}(t) \;=_{L.8.3.1}\; \mathtt{ms}(v_\mathsf{e})+\mathtt{ms}(t) \;>_{L.8.3.4}\; \mathtt{ms}(t).$$

<div style="text-align:right">□</div>

**Theorem 8.8** (Local termination). *Let* $a\in\{\mathtt{m},\mathtt{e}^{\mathtt{ms}},\mathtt{e}^{\mathtt{ss}},\mathtt{ms}_{\neg\multimap},\mathtt{ss}_{\neg\multimap}\}$. *Then* $\to_a$ *is strongly normalizing.*

*Proof.* For $\to_{\mathtt{e}^{\mathtt{ms}}}$ and $\to_{\mathtt{ms}_{\neg\multimap}}$ it is an immediate consequence of the fact that the measure decreases (Prop. 8.7) and that it is always positive (Lemma 8.3.4). The result is extended

to $\to_{\mathsf{e^{ss}}}$ and $\to_{\mathsf{ss}_{\neg\multimap}}$ by full composition (Prop. 7.1). For $\to_{\mathsf{m}}$, the statement is obvious, as at each step $\to_{\mathsf{m}}$ decreases the number of constructors. $\qquad\qquad\qquad\qquad\square$

The literature contains other partitions of linear logic cut elimination in strongly normalizing reductions: Danos [Dan90] and Joinet [Joi77] split it into axiom and non-axiom rewriting rules, Pagani and Tortora de Falco [PTdF10] into *structural* (a subset of exponential) and *logical* (i.e. non-structural). We are interested in the lolli/non-lolli partition because it mimics what happens in $\lambda$-calculi with ES, and also because of the future work described in *the next step* paragraph of the conclusions, for which the mentioned alternative partitions in the literature would not work.

**Next.** Three independent topics follow: confluence (Sect. 10), the good strategy (Sect. 11), and strong normalization (Sections 12 and 13). They can be read in any order. First, however, we need to develop some technical tools for dealing with contexts in proofs.

## 9. Some Technicalities

This section develops definitions and technical tools that are used in the proofs of the following sections to manage contexts, namely the outside-in order on contexts, contexts with *two* holes (called *double contexts*), and a technical but important *deformation lemma*. At a first (or even second) reading, the reader can skip this section.

**Definition 9.1** (Outside-in context order, disjoint contexts). We define the partial *outside-in* order $\prec$ over contexts as follows:

$$\frac{}{\langle\cdot\rangle \prec C} \qquad\qquad \frac{C \prec C'}{C''\langle C\rangle \prec C''\langle C'\rangle}$$

And say that $C$ is *outer* than $C'$ if $C \prec C'$. If $C \nprec C'$ and $C' \nprec C$ we say that $C$ and $C'$ are *disjoint*, and write $C \parallel C'$.

**Double Contexts.** Double contexts shall be used to compare two contexts on the same term. They have as base cases binary constructors (that is, tensor pairs, cuts, and subtractions) having contexts replacing their subterms, and as inductive cases they are simply closed by an ordinary context.

**Definition 9.2** (Double contexts). Double contexts $\mathbb{C}$ are defined by the following grammar.

$$\textsc{Double contexts} \quad \mathbb{C} \quad := \quad (C, C') \mid (\!|m\otimes C, x|\!)C' \mid [C{\to}x]C' \mid C\langle\mathbb{C}\rangle$$

Some easy facts about double contexts.

- *Plugging*: the plugging $\mathbb{C}\langle t, s\rangle$ of two terms $t$ and $s$ into a double context $\mathbb{C}$ is defined as expected and gives a term. The two ways of plugging one term $\mathbb{C}\langle t, \langle\cdot\rangle\rangle$ and $\mathbb{C}\langle\langle\cdot\rangle, s\rangle$ into a double context give instead a context. We often use $\mathbb{C}\langle t, \cdot\rangle$ and $\mathbb{C}\langle\cdot, s\rangle$ as compact notations for $\mathbb{C}\langle t, \langle\cdot\rangle\rangle$ and $\mathbb{C}\langle\langle\cdot\rangle, s\rangle$.
- *Pairs of disjoint positions and double contexts*: every pair of positions $C\langle t\rangle = C'\langle s\rangle$ which are disjoint, that is, such that $C \parallel C'$, gives rise to a double context $\mathbb{C}_{C,C'}$ such that $\mathbb{C}_{C,C'}\langle\cdot, s\rangle = C$ and $\mathbb{C}_{C,C'}\langle t, \cdot\rangle = C'$. Conversely, every double context $\mathbb{C}$ and every pair of terms $t$ and $s$ gives rise to two context $\mathbb{C}_l^{t,s} := \mathbb{C}\langle\cdot, s\rangle$ and $C_r^{t,s} := \mathbb{C}\langle t, \cdot\rangle$. Note that the two constructions are inverse of each other, that is, given $\mathbb{C}$, we have $\mathbb{C}_{\mathbb{C}_l^{t,s}, C_r^{t,s}} = \mathbb{C}$, and, conversely, given $C\langle t\rangle = C'\langle s\rangle$ we have $(\mathbb{C}_{C,C'})_l^{t,s} = C$ and $(\mathbb{C}_{C,C'})_r^{t,s} = C'$.

**Lemma 9.3.** $C\langle\mathbb{C}\langle t,\cdot\rangle\rangle = C\langle\mathbb{C}\rangle\langle t,\cdot\rangle$ *and* $C\langle\mathbb{C}\langle\cdot,t\rangle\rangle = C\langle\mathbb{C}\rangle\langle\cdot,t\rangle$.

*Proof.* By induction on $C$.  □

**Deformation Lemma.** We need a set of *deformation lemmas*, collected together in Lemma 9.4, that say what happens to a variable occurrence under the action of a rewriting step: for instance, given a step $M\langle\!\langle m\rangle\!\rangle \to_{\tt ms} s$, one lemma states that $s$ has shape $M'\langle\!\langle m\rangle\!\rangle$ unless the step reduces a cut containing $m$. The idea is that then $M'$ is the residual of $M$, but the statement(s) avoid using any form of residuals of steps, sub-terms, or constructors. The residual aspect is captured by specifying the preservation modulo a parameter—in the case mentioned above, this is obtained stating that there is a step $M\langle\!\langle v_{\tt m}\rangle\!\rangle \to_a M'\langle\!\langle v_{\tt m}\rangle\!\rangle$ for every $v_{\tt m}$.

The deformation lemmas shall be used in the following sections, namely in the proofs of local confluence (where preservation modulo a parameter is exploited), for the diamond property of the good strategy (for which the deformation lemmas shall have to be strengthened), and for the root cut expansion property (Prop. 12.2) at work in both the preservation of untyped strong normalization and typed normalization.

**Lemma 9.4** (Deformation properties). *Let* $\to\in\{\to_{\tt ss}, \to_{\tt ms}\}$.
(1) *Let* $C : t = M\langle\!\langle m\rangle\!\rangle \to_a s$ *and* $\to_a$ *be a* $\to$ *step. Then:*
  (a) *either* $s = M'\langle\!\langle m\rangle\!\rangle$ *for some* $M'$ *such that* $m \notin \mathtt{fv}(M')$ *and there is a step* $C'_{v_{\tt m}} :$ $M\langle\!\langle v_{\tt m}\rangle\!\rangle \to_a M'\langle\!\langle v_{\tt m}\rangle\!\rangle$ *for every* $v_{\tt m}$,
  (b) *or* $M = M'\langle[\langle\cdot\rangle\to n]u\rangle$ *for some* $M'$ *and* $C$ *reduces the cut on* $n$.
(2) *Let* $C : t = M\langle\!\langle t_m\rangle\!\rangle \to_a s$ *where* $t_m$ *is a par or a subtraction of conclusion* $m$, $M'$ *does not capture* $n$, *and* $\to_a$ *be a* $\to$ *step. Then* $s = M'\langle\!\langle s_m\rangle\!\rangle$ *where* $s_m$ *is an occurrence of* $m$ *of the same kind of* $t_m$, $M'$ *does not capture* $m$ *and* $m \notin \mathtt{fv}(M')$, *and there is a step* $C'_n : M\langle\!\langle\{n\to m\}t_m\rangle\!\rangle \to_a M'\langle\!\langle\{n\to m\}s_m\rangle\!\rangle$ *for every fresh multiplicative variable* $n$.
(3) *If* $C : M\langle(\!|m\,\mathfrak{B}\,x, y|\!)t\rangle \to_a s$ *is a* $\to$-*step and* $M$ *does not capture* $m$ *then* $s = M'\langle(\!|m\,\mathfrak{B}\,x, y|\!)t'\rangle$ *with* $M'$ *not capturing* $m$ *and such that there is a step* $C'_{C''} :$ $M\langle C''\langle t\rangle\rangle \to_a M'\langle C''\langle t'\rangle\rangle$ *for every context* $C''$ *capturing no more than* $x$ *and* $y$ *in* $t$ *and such that* $M$ *does not capture variables of* $C''$.
(4) *If* $C : M\langle(\!|m\otimes v, x|\!)r\rangle \to_a s$ *is a* $\to$-*step and* $M$ *does not capture* $m$ *then either*
  • $s = M'\langle(\!|m\otimes v, x|\!)r'\rangle$ *with* $C'_{C''} : M\langle C''\langle r\rangle\rangle \to_a M'\langle C''\langle r'\rangle\rangle$ *for every context* $C''$ *capturing no more than* $x$ *in* $r$ *and such that* $M$ *does not capture in* $C''$ *more variables than those in* $\mathtt{fv}(v)$, *or*
  • $s = M'\langle(\!|m\otimes v', x|\!)r\rangle$ *with* $C'_{C''} : M\langle C''\langle v\rangle\rangle \to_a M'\langle C''\langle v'\rangle\rangle$ *for every context* $C''$ *not capturing variables of* $v$ *and such that* $M$ *does not capture in* $C''$ *more variables than those in* $\mathtt{fv}(r) \setminus \{x\}$.
  *In both cases* $M'$ *does not capture* $m$.
(5) *Let* $C : t = C'\langle\!\langle e\rangle\!\rangle \to_a s$ *and* $\to_a$ *be a* $\to_{\tt ms}$ *step. Then:*
  (a) *either* $s = C''\langle\!\langle e\rangle\!\rangle$ *for some* $C''$ *such that there is a step* $C^{\bullet}_{v_{\tt e}} : C'\langle\!\langle v_{\tt e}\rangle\!\rangle \to_a C''\langle\!\langle v_{\tt e}\rangle\!\rangle$ *for every* $v_{\tt e}$,
  (b) *or* $C' = C''\langle[C'''\to f]u\rangle$ *for some* $C''$ *and* $C'''$, *and* $C$ *reduces a redex where the active cut is the one on* $f$.
(6) *Let* $C : t = C'\langle(\!|e?x|\!)u\rangle \to_a s$ *with* $C'$ *not capturing* $e$ *and* $\to_a$ *be a* $\to_{\tt ms}$ *step. Then:*
  (a) *either* $s = C''\langle(\!|e?x|\!)u'\rangle$ *for some* $C''$ *such that there is a step* $C^{\bullet}_{C''} : C'\langle C''\langle u\rangle\rangle \to_a$ $C''\langle C''\langle u'\rangle\rangle$ *for every* $C''$ *capturing no more than* $x$ *in* $u$ *and* $u'$,

(b) *or $C' = C''\langle[C'''{\to}f]u\rangle$ for some $C''$ and $C'''$, and $C$ reduces a redex given by the cut on $f$.*

*Proof.* The proof is by induction on the rewriting step in each point, and is a tedious check. In [Acc22], we give all the (many!) details for the first deformation property. The proofs of the others are minor variations. □

## 10. Untyped Confluence

Here we prove confluence for the untyped ESC, using an elegant technique based on local diagrams and local termination. The technique is the Hindley-Rosen method. In our case, it amounts to prove that the multiplicative and exponential rules $\to_{\mathtt{m}}$ and $\to_{\mathtt{e}^{\mathtt{ms}}}$ are confluent separately, proved by local termination and Newman lemma, and commute, proved by local termination and Hindley's strong commutation. Confluence then follows by Hindley-Rosen lemma, for which the union of confluent and commuting reductions is confluent.

The Hindley-Rosen method is a modular technique often used for confluence of extensions of the $\lambda$-calculus, for instance in [AD17, FR19, Sau08, CG14, Rév92, BKR21, AFM$^+$95, AP12]. It is also used for untyped proof nets by Pagani and Tortora de Falco [PTdF10]. Confluence for untyped proof nets is also proved by Danos [Dan90] and Tranquilli [Tra09] via finite developments. Essentially, local termination internalizes finite developments.

**The Glitch.** For the untyped ESC there is a slight flaw due to clashes. The following local diagram, indeed, can be closed only if $[v{\to}m]$ is not a clashing cut (precisely, when $v$ is not an exponential value nor an abstraction), or with cut equivalence $\equiv_{\mathtt{cut}}$:

$$[v{\to}m]M'\langle[m{\to}n]M\langle(\!|n\,\mathfrak{R}\,x,y|\!)t\rangle\rangle \quad \to_{\mathtt{axm}_1} \quad M'\langle[v{\to}n]M\langle(\!|n\,\mathfrak{R}\,x,y|\!)t\rangle\rangle$$
$$\downarrow_{\mathtt{axm}_2}$$
$$[v{\to}m]M'\langle M\langle(\!|m\,\mathfrak{R}\,x,y|\!)t\rangle\rangle$$

A second similarly problematic diagram is obtained by replacing the par with a subtraction.

Now, if there is a clash, the fact that confluence holds only up to $\equiv_{\mathtt{cut}}$ is irrelevant: the clash is a bigger issue. Moreover:

(1) *(Recursive) types remove clashes:* clashes are ruled out by our typing (Lemma 7.6), but also by recursive types such as those used for typing the untyped $\lambda$-calculus, or even the more general weak recursive typing considered by Ehrhard and Regnier in [ER06].
(2) *Proof nets are not better:* because of weakenings, IMELL proof nets need *jumps* and jump *rewiring rules*. Confluence should then be proved *up to jump rewiring*, which is analogous to confluence up to $\equiv_{\mathtt{cut}}$.
(3) *Good steps are glitch-free:* anticipating from the next section, the glitch does not affect *good steps*, as the $\mathtt{axm}_1$ steps in the diagram above is *bad*.

**The Proof.** We first prove the local properties.

**Proposition 10.1** (Local confluence)**.**
(1) Multiplicative clash-free diamond*: $\to_{\mathtt{m}}$ is diamond on clash-free terms.*
(2) Exponential local confluence*: $\to_{\mathtt{e}^{\mathtt{ss}}}$ and $\to_{\mathtt{e}^{\mathtt{ms}}}$ are both locally confluent.*

*Proof.*

(1) The formal statement is: $t_1 \; {}_{\mathtt{m}}\!\!\leftarrow t_0 \to_{\mathtt{m}} t_2$ with $t_1 \neq t_2$ implies that there exists $t_3$ such that $t_1 \to_{\mathtt{m}} t_3 \; {}_{\mathtt{m}}\!\!\leftarrow t_2$. Let $a, b \in \{\mathtt{axm}_2, \mathtt{axm}_1, \otimes, \multimap\}$, $t_0 \to_a t_1$, and $t_0 \to_b t_2$. The proof is by induction on $t_0 \to_a t_1$, that is, by induction on the context $C$ closing the root rule $\mapsto_a$, and case analysis of $t_0 \to_b t_2$. Here we show only the case for $\mapsto_{\mathtt{axm}_1}$, which is the most interesting one, requiring the deformation lemma and allowing us to discuss the case of clashes. The other cases are in [Acc22].

We have $t_0 = [v_{\mathtt{m}} \to m] M \langle\!\langle m \rangle\!\rangle \mapsto_{\mathtt{axm}_1} M \langle\!\langle v_{\mathtt{m}} \rangle\!\rangle = t_1$. If $\to_b$ takes place entirely in $v_{\mathtt{m}}$ then the diagram closes in one step on both sides. If $\to_b$ takes place in $M \langle\!\langle m \rangle\!\rangle$, that is, $M \langle\!\langle m \rangle\!\rangle \to_b s$ then by deformation (Lemma 9.4.1) there are two mutually exclusive cases:

(a) $s = M' \langle\!\langle m \rangle\!\rangle$ for some $M'$ such that $m \notin \mathtt{fv}(M')$ and $M \langle\!\langle v_{\mathtt{m}} \rangle\!\rangle \to_b M' \langle\!\langle v_{\mathtt{m}} \rangle\!\rangle$ for every $v_{\mathtt{m}}$. Then:

$$
\begin{array}{ccc}
[v_{\mathtt{m}} \to m] M \langle\!\langle m \rangle\!\rangle & \mapsto_{\mathtt{axm}_1} & M \langle\!\langle v_{\mathtt{m}} \rangle\!\rangle \\
\downarrow b & & \downarrow b \\
[v_{\mathtt{m}} \to m] M' \langle\!\langle m \rangle\!\rangle & \mapsto_{\mathtt{axm}_1} & M' \langle\!\langle v_{\mathtt{m}} \rangle\!\rangle
\end{array}
$$

(b) $M = M' \langle [\langle \cdot \rangle \to n] u \rangle$ and $\to_b$ reduces the cut on $n$, which then is a $\to_{\mathtt{axm}_2}$ step. Then $u = M'' \langle o_n \rangle$ and diagram depends on the occurrence $o_n$ that interacts with the cut:

- *Variable*, that is, $o_n = n$. Then:

$$
\begin{array}{ccc}
[v_{\mathtt{m}} \to m] M' \langle [m \to n] M'' \langle n \rangle \rangle & \mapsto_{\mathtt{axm}_1} & M' \langle [v_{\mathtt{m}} \to n] M'' \langle n \rangle \rangle \\
\downarrow \mathtt{axm}_2 & & \downarrow \mathtt{axm}_1 \\
[v_{\mathtt{m}} \to m] M' \langle M'' \langle m \rangle \rangle & \mapsto_{\mathtt{axm}_1} & M' \langle M'' \langle v_{\mathtt{m}} \rangle \rangle
\end{array}
$$

Note a curious fact: the diagram turns a $\to_{\mathtt{axm}_2}$ step into a $\to_{\mathtt{axm}_1}$ step, that is, $\to_{\mathtt{axm}_2}$ and $\to_{\mathtt{axm}_1}$ do not commute. Note however that the $\to_{\mathtt{axm}_2}$ step can also be seen as a $\to_{\mathtt{axm}_1}$ step, as the two rules superpose in this case.

- *Par*, that is, $o_n = (\!| n \,\otimes\!\!\!\! x, y |\!) r$, and the span is:

$$
\begin{array}{ccc}
[v_{\mathtt{m}} \to m] M' \langle [m \to n] M'' \langle (\!| n \,\otimes\!\!\!\! x, y |\!) r \rangle \rangle & \mapsto_{\mathtt{axm}_1} & M' \langle [v_{\mathtt{m}} \to n] M'' \langle (\!| n \,\otimes\!\!\!\! x, y |\!) r \rangle \rangle \\
\downarrow \mathtt{axm}_2 & & \\
[v_{\mathtt{m}} \to m] M' \langle M'' \langle (\!| m \,\otimes\!\!\!\! x, y |\!) r \rangle \rangle & &
\end{array}
$$

By clash-freeness, $v_{\mathtt{m}} = (L \langle v \rangle, L' \langle v' \rangle)$, and the diagram is:

$$
\begin{array}{ccc}
[(L \langle v \rangle, L' \langle v' \rangle) \to m] M' \langle [m \to n] M'' \langle (\!| n \,\otimes\!\!\!\! x, y |\!) r \rangle \rangle & \mapsto_{\mathtt{axm}_1} & M' \langle [(L \langle v \rangle, L' \langle v' \rangle) \to n] M'' \langle (\!| n \,\otimes\!\!\!\! x, y |\!) r \rangle \rangle \\
\downarrow \mathtt{axm}_2 & & \downarrow \otimes \\
[(L \langle v \rangle, L' \langle v' \rangle) \to m] M' \langle M'' \langle (\!| m \,\otimes\!\!\!\! x, y |\!) r \rangle \rangle & \mapsto_{\otimes} & M' \langle M'' \langle L \langle [v \to x] L' \langle [v' \to y] r \rangle \rangle \rangle
\end{array}
$$

Note that the diagram closes using $\to_{\otimes}$ rather than $\to_{\mathtt{axm}_2}$ or $\to_{\mathtt{axm}_1}$. In absence of clash-freeness, note that the diagram closes also with $\equiv_{\mathtt{cut}}$.

- *Subtraction*, that is, $o_n = (\!| n \oslash v, x |\!) r$, and the span is:

$$
\begin{array}{ccc}
[v_{\mathtt{m}} \to m] M' \langle [m \to n] M'' \langle (\!| n \oslash v, x |\!) r \rangle \rangle & \mapsto_{\mathtt{axm}_1} & M' \langle [v_{\mathtt{m}} \to n] M'' \langle (\!| n \oslash v, x |\!) r \rangle \rangle \\
\downarrow \mathtt{axm}_2 & & \\
[v_{\mathtt{m}} \to m] M' \langle M'' \langle (\!| m \oslash v, x |\!) r \rangle \rangle & &
\end{array}
$$

By clash-freeness, $v_{\mathtt{m}} = \lambda y. L \langle v' \rangle$, and the diagram is:

$$
\begin{array}{ccc}
[\lambda y. L \langle v' \rangle \to m] M' \langle [m \to n] M'' \langle (\!| n \oslash v, x |\!) r \rangle \rangle & \mapsto_{\mathtt{axm}_1} & M' \langle [\lambda y. L \langle v' \rangle \to n] M'' \langle (\!| n \oslash v, x |\!) r \rangle \rangle \\
\downarrow \mathtt{axm}_2 & & \downarrow \multimap \\
[\lambda y. L \langle v' \rangle \to m] M' \langle M'' \langle (\!| m \oslash v, x |\!) r \rangle \rangle & \mapsto_{\multimap} & M' \langle M'' \langle [v \to y] L \langle [v' \to x] r \rangle \rangle \rangle
\end{array}
$$

Note that the diagram closes using $\to_{-\circ}$ rather than $\to_{\mathtt{axm}_2}$ or $\to_{\mathtt{axm}_1}$. In absence of clash-freeness, note that the diagram closes also with $\equiv_{\mathtt{cut}}$.

(2) The proof follows the same structure as for $\to_{\mathtt{m}}$. Again, we show only the root case of $\to_{\mathtt{axe}_1}$, that is, $\mapsto_{\mathtt{axe}_1}$, which is the most interesting one. The other cases are in [Acc22]. We have $t_0 = [v_{\mathsf{e}}{\to}e]C\langle\!\langle e\rangle\!\rangle \mapsto_{\mathtt{axe}_1} [v_{\mathsf{e}}{\to}e]C\langle\!\langle v_{\mathsf{e}}\rangle\!\rangle = t_1$. Three cases:

(a) $\to_b$ *takes place in* $v_{\mathsf{e}}$. Then the diagram is (note that it is not a diamond diagram):

$$[v_{\mathsf{e}}{\to}e]C\langle\!\langle e\rangle\!\rangle \quad\mapsto_{\mathtt{axe}_1}\quad [v_{\mathsf{e}}{\to}e]C\langle\!\langle v_{\mathsf{e}}\rangle\!\rangle$$
$$\downarrow b$$
$$[v'_{\mathsf{e}}{\to}e]C\langle\!\langle v_{\mathsf{e}}\rangle\!\rangle$$
$$\downarrow b \qquad\qquad\qquad\qquad\qquad \downarrow b$$
$$[v'_{\mathsf{e}}{\to}e]C\langle\!\langle e\rangle\!\rangle \quad\mapsto_{\mathtt{axe}_1}\quad [v'_{\mathsf{e}}{\to}e]C\langle\!\langle v'_{\mathsf{e}}\rangle\!\rangle$$

(b) $\to_b$ *takes place in* $C\langle e\rangle$. Then by the deformation lemma (Lemma 9.4.5) there are two mutually exclusive cases:

- $s = C'\langle\!\langle e\rangle\!\rangle$ for some $C'$ such that $C\langle\!\langle v_{\mathsf{e}}\rangle\!\rangle \to_a C'\langle\!\langle v_{\mathsf{e}}\rangle\!\rangle$ for every $v_{\mathsf{e}}$. Then:

$$[v_{\mathsf{e}}{\to}e]C\langle\!\langle e\rangle\!\rangle \quad\mapsto_{\mathtt{axe}_1}\quad [v_{\mathsf{e}}{\to}e]C\langle\!\langle v_{\mathsf{e}}\rangle\!\rangle$$
$$\downarrow b \qquad\qquad\qquad\qquad\qquad \downarrow b$$
$$[v_{\mathsf{e}}{\to}e]C'\langle\!\langle e\rangle\!\rangle \quad\mapsto_{\mathtt{axe}_1}\quad [v_{\mathsf{e}}{\to}e]C'\langle\!\langle v_{\mathsf{e}}\rangle\!\rangle$$

- or $C = C'\langle[C''{\to}f]u\rangle$ for some $C'$ and $C''$, and $\to_b$ reduces a redex given by the cut on $f$. Cases of the reduced redex:

  - $\to_{\mathtt{axe}_1}$. Then we have $C\langle\!\langle e\rangle\!\rangle = C'\langle[e{\to}f]C'''\langle\!\langle f\rangle\!\rangle\rangle$ for some $C'''$ (and $C''$ is empty), and the diagram closes as follows (it is not diamond):

$$[v_{\mathsf{e}}{\to}e]C'\langle[e{\to}f]C'''\langle\!\langle f\rangle\!\rangle\rangle \quad\mapsto_{\mathtt{axe}_1}\quad [v_{\mathsf{e}}{\to}e]C'\langle[v_{\mathsf{e}}{\to}f]C'''\langle\!\langle f\rangle\!\rangle\rangle$$
$$\downarrow_{\mathtt{axe}_1}$$
$$[v_{\mathsf{e}}{\to}e]C'\langle[v_{\mathsf{e}}{\to}f]C'''\langle\!\langle v_{\mathsf{e}}\rangle\!\rangle\rangle$$
$$\downarrow_{\mathtt{axe}_1} \qquad\qquad\qquad\qquad\qquad \uparrow_{\mathtt{axe}_1}$$
$$[v_{\mathsf{e}}{\to}e]C'\langle[e{\to}f]C'''\langle\!\langle e\rangle\!\rangle\rangle \quad\mapsto_{\mathtt{axe}_1}\quad [v_{\mathsf{e}}{\to}e]C'\langle[v_{\mathsf{e}}{\to}f]C'''\langle\!\langle e\rangle\!\rangle\rangle$$

  - $\to_!$. Then we have $C\langle\!\langle e\rangle\!\rangle = C'\langle[!r{\to}f]C'''\langle\!\langle (\!|f?x|\!)u\rangle\!\rangle\rangle$ for some $C'''$ and with $e$ occurring in $r$. To write down the rewriting step we have to decompose $r$ as a left context and a value, that is, $r = L\langle v\rangle$. Now, $e$ can be in either $L$ or $v$. We consider the latter case, which is easier to spell out in symbols, the former case is analogous, just heavier to write down. Then assume that $v = C''\langle\!\langle e\rangle\!\rangle$, so that $r = L\langle C''\langle\!\langle e\rangle\!\rangle\rangle$. Then we have the following (non-diamond) diagram (which we have to write vertically because it does not fit, note that the top and bottom term are the same):

$$[v_{\mathsf{e}}{\to}e]C'\langle[!L\langle C''\langle\!\langle e\rangle\!\rangle\rangle{\to}f]C'''\langle\!\langle (\!|f?x|\!)u\rangle\!\rangle\rangle$$
$$\downarrow_{\mathtt{axe}_1}$$
$$[v_{\mathsf{e}}{\to}e]C'\langle[!L\langle C''\langle\!\langle v_{\mathsf{e}}\rangle\!\rangle\rangle{\to}f]C'''\langle\!\langle (\!|f?x|\!)u\rangle\!\rangle\rangle$$
$$\downarrow_!$$
$$[v_{\mathsf{e}}{\to}e]C'\langle[!L\langle C''\langle\!\langle v_{\mathsf{e}}\rangle\!\rangle\rangle{\to}f]C'''\langle\!\langle L\langle[C''\langle\!\langle e\rangle\!\rangle{\to}x]u\rangle\rangle\!\rangle\rangle$$
$$\uparrow_{\mathtt{axe}_1}$$
$$[v_{\mathsf{e}}{\to}e]C'\langle[!L\langle C''\langle\!\langle e\rangle\!\rangle\rangle{\to}f]C'''\langle\!\langle L\langle[C''\langle\!\langle e\rangle\!\rangle{\to}x]u\rangle\rangle\!\rangle\rangle$$
$$\uparrow_{\mathtt{axe}_1}$$
$$[v_{\mathsf{e}}{\to}e]C'\langle[!L\langle C''\langle\!\langle v_{\mathsf{e}}\rangle\!\rangle\rangle{\to}f]C'''\langle\!\langle L\langle[C''\langle\!\langle v_{\mathsf{e}}\rangle\!\rangle{\to}x]u\rangle\rangle\!\rangle\rangle$$
$$\uparrow_!$$
$$[v_{\mathsf{e}}{\to}e]C'\langle[!L\langle C''\langle\!\langle e\rangle\!\rangle\rangle{\to}f]C'''\langle\!\langle (\!|f?x|\!)u\rangle\!\rangle\rangle$$

  - $\to_{\mathtt{w}}$. Then $C\langle\!\langle e\rangle\!\rangle = C'\langle[C''\langle e\rangle{\to}f]s\rangle \to_{\mathtt{w}} C'\langle s\rangle$ and we have the following (non-diamond) diagram:

$$[v_{\mathsf{e}}{\to}e]C'\langle[C''\langle e\rangle{\to}f]s\rangle \quad \mapsto_{\mathtt{axe_1}} \quad [v_{\mathsf{e}}{\to}e]C'\langle[C''\langle v_{\mathsf{e}}\rangle{\to}f]s\rangle$$
$$\downarrow_{\mathtt{w}} \qquad\qquad\qquad\qquad\qquad \downarrow_{\mathtt{w}}$$
$$[v_{\mathsf{e}}{\to}e]C'\langle s\rangle \qquad = \qquad [v_{\mathsf{e}}{\to}e]C'\langle s\rangle$$

(c) $\to_b$ *involves the same root cut of* $\to_a$. Then $t_0 = [f{\to}e]C\langle\!\langle e\rangle\!\rangle = [f{\to}e]C'\langle t_e\rangle$ for a context $C' \neq C$ that does not capture $e$. Two sub-cases:

- *Variable*, that is, $t_e = e$. Then $\to_b$ is also a $\mapsto_{\mathtt{axe_1}}$ step. Note that $C$ and $C'$ are disjoint, that is, $C \parallel C'$, so that there is a double context $\mathbb{C}_{C,C'}$ such that $\mathbb{C}_{C,C'}\langle\cdot,e\rangle = C$ and $\mathbb{C}_{C,C'}\langle e,\cdot\rangle = C'$. Then $t_0 = [v_{\mathsf{e}}{\to}e]\mathbb{C}_{C,C'}\langle e,e\rangle$ and
$$[v_{\mathsf{e}}{\to}e]\mathbb{C}_{C,C'}\langle e,e\rangle \quad \mapsto_{\mathtt{axe_1}} \quad [v_{\mathsf{e}}{\to}e]\mathbb{C}_{C,C'}\langle v_{\mathsf{e}},e\rangle$$
$$\downarrow_{\mathtt{axe_1}} \qquad\qquad\qquad\qquad \downarrow_{\mathtt{axe_1}}$$
$$[v_{\mathsf{e}}{\to}e]\mathbb{C}_{C,C'}\langle e,v_{\mathsf{e}}\rangle \quad \mapsto_{\mathtt{axe_1}} \quad [v_{\mathsf{e}}{\to}e]\mathbb{C}_{C,C'}\langle v_{\mathsf{e}},v_{\mathsf{e}}\rangle$$

- *Dereliction*, that is, $t_e = (\!|e?x|\!)s$. If $C$ and $C'$ are disjoint then the reasoning goes as if $t_e$ were a variable. Otherwise, $C' \prec C$, that is, there exists $C''$ such that $C = C'\langle(\!|e?x|\!)C''\rangle$. Now, the the diagram depends on what kind of value is $v_{\mathsf{e}}$. For the sake of simplicity, we deal with the case $v_{\mathsf{e}} = f$. Then:
$$[f{\to}e]C'\langle(\!|e?x|\!)C''\langle e\rangle\rangle \quad \mapsto_{\mathtt{axe_1}} \quad [f{\to}e]C'\langle(\!|e?x|\!)C''\langle f\rangle\rangle$$
$$\downarrow_{\mathtt{axe_2}} \qquad\qquad\qquad\qquad \downarrow_{\mathtt{axe_2}}$$
$$[f{\to}e]C'\langle(\!|f?x|\!)C''\langle e\rangle\rangle \quad \mapsto_{\mathtt{axe_1}} \quad [f{\to}e]C'\langle(\!|f?x|\!)C''\langle f\rangle\rangle$$
If $v_{\mathsf{e}} = {!}s$ then there is a similar diagram where $\mapsto_{\mathtt{axe_2}}$ steps become $\mapsto_{!}$ steps. $\square$

In contrast to confluence, commutation of two reductions $\to_1$ and $\to_2$ does *not* follow from their *local* commutation and strong normalization. In our case however the rules verify Hindley's *strong* (local) commutation [Hin64], for which $s_{1\ 1}{\leftarrow}t \to_2 s_2$ implies that there exists $u$ such that $s_1 \to_2^* u$ and either $u_{\ 1}{\leftarrow}s_2$ or $u = s_2$, that is, on one side of the commutation there are no duplications of steps. Strong commutation and strong normalization do imply commutation.

**Proposition 10.2** (Strong commutation). $\to_{\mathsf{e^{ss}}}$ *and* $\to_{\mathsf{e^{ms}}}$ *both strongly commute over* $\to_{\mathsf{m}}$.

*Proof.* The proof is an analysis of cases very similar to the one for local confluence. Some details are in [Acc22]. $\square$

Then we lift the local properties using local termination and conclude using Hindley-Rosen lemma.

**Theorem 10.3** (Confluence). *The relations* $\to_{\mathsf{ss}}$ *and* $\to_{\mathsf{ms}}$ *are confluent on clash-free terms.*

*Proof.* We deal with $\to_{\mathsf{ms}}$, the reasoning for $\to_{\mathsf{e^{ss}}}$ is identical. By Newman lemma, local confluence (Prop. 10.1) and local termination (Theorem 8.8) imply that $\to_{\mathsf{m}}$ and $\to_{\mathsf{e^{ss}}}$ are confluent separately. By a result of Hindley [Hin64], strong commutation (Prop. 10.2) and local termination imply that $\to_{\mathsf{m}}$ and $\to_{\mathsf{e^{ss}}}$ commute. By Hindley-Rosen lemma, $\to_{\mathsf{ms}} = \to_{\mathsf{m}} \cup \to_{\mathsf{e^{ms}}}$ is confluent. $\square$

As it was mentioned at the end of Sect. 5, our micro-step rules for exponential axioms are different from the one at work in proof nets. The proof net micro-step rule, when formulated on terms, looks as a small-step rule:

$$[f{\to}e]t \quad \to \quad \{f{\to}e\}t$$

If considered with our other micro-step rules, in particular our $\to_{!}$, it generates a closable but unpleasant local confluence diagram:

DOMINATING FREE VARIABLES OF CONTEXTS

$$\mathtt{dv}(\langle\cdot\rangle) := \emptyset$$
$$\mathtt{dv}((C,v)) := \mathtt{dv}(C)$$
$$\mathtt{dv}((v,C)) := \mathtt{dv}(C)$$
$$\mathtt{dv}(\lambda x.C) := \mathtt{dv}(C) \setminus \{x\}$$
$$\mathtt{dv}(!C) := \mathtt{dv}(C)$$
$$\mathtt{dv}([v{\to}x]C) := \mathtt{dv}(C) \setminus \{x\}$$
$$\mathtt{dv}([V{\to}x]t) := \mathtt{dv}(V)$$

$$\mathtt{dv}((\!|m\,\mathfrak{R}\,x,y|\!)C) := \begin{cases} \{m\} \cup (\mathtt{dv}(C) \setminus \{x,y\}) & \text{if } x \in \mathtt{dv}(C) \\ & \qquad \text{or } y \in \mathtt{dv}(C) \\ \mathtt{dv}(C) & \text{otherwise.} \end{cases}$$

$$\mathtt{dv}((\!|m\,\oslash\,V,x|\!)t) := \{m\} \cup \mathtt{dv}(V)$$

$$\mathtt{dv}((\!|m\,\oslash\,v,x|\!)C) := \begin{cases} \{m\} \cup (\mathtt{dv}(C) \setminus \{x\}) & \text{if } x \in \mathtt{dv}(C) \\ \mathtt{dv}(C) & \text{otherwise.} \end{cases}$$

$$\mathtt{dv}((\!|e?x|\!)C) := \begin{cases} \{e\} \cup (\mathtt{dv}(C) \setminus \{x\}) & \text{if } x \in \mathtt{dv}(C) \\ \mathtt{dv}(C) & \text{otherwise.} \end{cases}$$

GOOD VALUE CONTEXTS
$$V_G ::= \langle\cdot\rangle \mid (G,t) \mid (t,G) \mid \lambda x.G \mid !G$$

GOOD CONTEXTS
$$G ::= V_G \mid (\!|m\,\mathfrak{R}\,x,y|\!)G \mid (\!|m\,\oslash\,v,x|\!)G \mid (\!|m\,\oslash\,V_G,x|\!)t \mid (\!|e?x|\!)G \mid [v{\to}x]G \text{ if } x \notin \mathtt{dv}(G)$$

BAD CONTEXTS
$$B ::= [V{\to}x]t \mid [v{\to}x]C \text{ if } x \in \mathtt{dv}(C) \mid C\langle B\rangle$$

Figure 5: Definitions for the good strategy: dominating free variables, good and bad contexts.

$$\begin{array}{ccccc}
[!v{\to}f]C\langle[f{\to}e]t\rangle & \mapsto_! & [!v{\to}f]C\langle[!v{\to}e]t\rangle & \dashrightarrow^+_{\mathsf{ems}} & [!v{\to}f]C\langle\{!v{\to}e\}t\rangle \\
\downarrow & & & & \downarrow_{\mathsf{ems}+} \\
[!v{\to}f]C\langle\{f{\to}e\}t\rangle & \dashrightarrow^+_{\mathsf{ems}} & \{!v{\to}f\}C\langle\{f{\to}e\}t\rangle & = & \{!v{\to}f\}C\langle\{!v{\to}e\}t\rangle
\end{array}$$

In order to close the local diagram one needs to *fully develop* some exponential cuts (note $\dashrightarrow^+_{\mathsf{ems}}$), which is a small-step concept and not a micro-step one. This is why we adopt different axiom rules, for which local confluence diagrams never need small-step developments.

## 11. THE GOOD STRATEGY

In this section, we define the *good cut elimination strategy* $\to_{\mathsf{G}}$ for the untyped ESC, show various of its properties, including the sub-term property, and prove that it provides a polynomial cost model.

**Breaking the Sub-Term Property.** When does the sub-term property not hold? One has to duplicate an exponential value $v_{\mathsf{e}}$ *touched* by previous steps. In our setting, *touched* can mean two things. Either a redex *fully* contained in $v_{\mathsf{e}}$ is reduced, obtaining $v'_{\mathsf{e}}$, and then $v'_{\mathsf{e}}$ is duplicated (or erased), as in the step marked with ★ in the following diagram (the other, dashed path of which has the sub-term property):

$$\begin{array}{ccc}
[v_{\mathsf{e}}{\to}e]C\langle\!\langle e\rangle\!\rangle & \to_{\mathsf{ms}} & [v'_{\mathsf{e}}{\to}e]C\langle\!\langle e\rangle\!\rangle \\
\downarrow_{\mathsf{axe}_1} & & \downarrow_{\mathsf{axe}_1} \text{ ★} \\
[v_{\mathsf{e}}{\to}e]C\langle\!\langle v_{\mathsf{e}}\rangle\!\rangle \dashrightarrow_{\mathsf{ms}} [v'_{\mathsf{e}}{\to}e]C\langle\!\langle v_{\mathsf{e}}\rangle\!\rangle & \dashrightarrow_{\mathsf{ms}} & [v'_{\mathsf{e}}{\to}e]C\langle\!\langle v'_{\mathsf{e}}\rangle\!\rangle
\end{array}$$

The other way of *touching* $v_\mathsf{e}$ is when a cut $[v'_\mathsf{e} \to e]$ *external* to $v_\mathsf{e}$ acts on some exponential variable $e$ in $v_\mathsf{e} = V\langle\!\langle e \rangle\!\rangle$:

$$[v'_\mathsf{e} \to e]C\langle [V\langle\!\langle e \rangle\!\rangle \to f]C'\langle\!\langle f \rangle\!\rangle\rangle \qquad\qquad \mapsto_{\mathtt{axe}_1} \qquad\qquad [v'_\mathsf{e} \to e]C\langle [V\langle\!\langle v'_\mathsf{e} \rangle\!\rangle \to f]C'\langle\!\langle f \rangle\!\rangle\rangle$$

$$\downarrow_{\mathtt{axe}_1} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow_{\mathtt{axe}_1} \,\,\bigstar$$

$$[v'_\mathsf{e} \to e]C\langle [V\langle\!\langle e \rangle\!\rangle \to f]C'\langle\!\langle V\langle\!\langle e \rangle\!\rangle \rangle\!\rangle\rangle \dashrightarrow_{\mathtt{axe}_1} [v'_\mathsf{e} \to e]C\langle [V\langle\!\langle v'_\mathsf{e} \rangle\!\rangle \to f]C'\langle\!\langle V\langle\!\langle e \rangle\!\rangle \rangle\!\rangle\rangle \dashrightarrow_{\mathtt{axe}_1} [v'_\mathsf{e} \to e]C\langle [V\langle\!\langle v'_\mathsf{e} \rangle\!\rangle \to f]C'\langle\!\langle V\langle\!\langle v'_\mathsf{e} \rangle\!\rangle \rangle\!\rangle\rangle$$

Similar diagrams can be obtained using $\to_!$ rather than $\to_{\mathtt{axe}_1}$, and these are all the local confluence diagrams for $\to_{\mathtt{ms}}$ that are not squares nor triangles (see Prop. 10.1 and Prop. 10.2).

Preventing these situations from happening, thus forcing evaluation to follow the other side of the diagram, is easy. It is enough to forbid the position of the redex to be inside the left sub-term of a cut—we say inside a *cut value* for short. It is however not enough, because cuts are also *created*. Consider:

$$[\lambda e.(e, e) \to m](\!| m \otimes v_\mathsf{e}, x |\!)t \to_{\mathtt{ms}} [\lambda e.(e, e) \to m](\!| m \otimes v'_\mathsf{e}, x |\!)t \to_{\multimap}$$
$$[v'_\mathsf{e} \to e][(e, e) \to x]t \xrightarrow{\bigstar}_{\mathtt{axe}_1} [v'_\mathsf{e} \to e][(v'_\mathsf{e}, e) \to x]t$$

Reducing inside the subtraction value $v_\mathsf{e}$ leads to a *later* breaking of the sub-term property by the $\mathtt{axe}_1$ step, because the $\to_{\multimap}$ step creates a cut with $v'_\mathsf{e}$ inside. Preventing these cases is tricky, because forbidding reducing subtraction values leads to cut elimination stopping too soon, without producing a cut-free term. In the $\lambda$-calculus, it corresponds to forbidding reducing inside arguments, which leads to *head* reduction, that does not compute normal $\lambda$-terms. We shall then forbid reducing only subtraction values which are *at risk* of becoming cuts. It might look like the dangerous positions are those in $v$ in $[v_\mathsf{m} \to m]M\langle(\!| m \otimes v, x |\!)t \rangle$, i.e. those where the surrounding subtraction is involved in a cut. They are in fact more general. Consider the positions inside $v$ in $t$ here:

$$t \;=\; [!v_\mathsf{m} \to e](\!| e?m |\!)(\!| m \otimes v, x |\!)s \;\to_!\; [!v_\mathsf{m} \to e][v_\mathsf{m} \to m](\!| m \otimes v, x |\!)s$$

they are also dangerous, because they reduce to those of the previous kind. Therefore, we need to ensure that the conclusion $m$ of the subtraction is not *hereditarely* involved in a cut. The key notion is the one of *dominating (free) variables* $\mathtt{dv}(C)$ of a context (where $C$ is meant to be the position of a redex), in Fig. 5, the base case of which is for $(\!| m \otimes V, x |\!)t$. If $C$ is a position and $x \in \mathtt{dv}(C)$ then $[v \to x]C$ turns $C$ into a dangerous position, that is, a redex of position $[v \to x]C : t \to_{\mathtt{ms}} s$ might lead to a breaking of the sub-term property later on during cut elimination. In the last example, $e$ dominates the context $(\!| e?m |\!)(\!| m \otimes V, x |\!)s$ of $(\!| e?m |\!)(\!| m \otimes v, x |\!)s$ for every $V$.

**Good and Bad Contexts and Steps.** The previous considerations lead to the notions of *good* and *bad contexts* in Fig. 5. A good context forbids the two ways of breaking the sub-term property: its hole cannot be in a cut value (note the absence of the production $[V_G \to x]t$) nor in a subtraction value such that one of its dominating variables is cut (because of the production $[v \to x]G$ if $x \notin \mathtt{dv}(G)$). Every other step is allowed. The next lemma ensures that bad and good contexts are indeed complementary concepts.

**Lemma 11.1** (Good/bad partition). *Let $C$ be a context. Then $C$ is either good or bad.*

*Proof.* By induction on $C$. The empty context $\langle \cdot \rangle$ is good and not bad. For all inductive cases but cut, it follows from the *i.h.* The cut cases:

- $C = [V \to x]t$. Then $C$ is bad and not good.

- $C = [v{\to}x]C'$. By *i.h.*, $C'$ is either good or bad. If $C'$ is bad then $C$ is bad, and $C$ is not good because $C'$ is not good. If $C'$ is good, consider whether $x \in \mathtt{dv}(C')$. If it does, then $C$ is bad and not good. If instead $x \notin \mathtt{dv}(C')$ then $C$ is good and not bad. □

**Definition 11.2** (Good/bad step, good strategy). A micro step $C : t \to_{\mathtt{ms}} s$ is *good* if its position $C$ is good. In such a case, we write $t \to_{\mathtt{G}} s$. The *good cut elimination strategy* is simply $\to_{\mathtt{G}}$. A step $t \to_{\mathtt{ms}} s$ is *bad* if it is not good, and we then write $t \to_{\mathtt{B}} s$. We also use $\to_{\mathtt{G}a}$ and $\to_{\mathtt{B}a}$ to stress that the good/bad step is of kind $a \in \{\mathtt{axm}_1, \mathtt{axm}_2, \otimes, \multimap, \mathtt{axe}_1, \mathtt{axe}_2, !, \mathtt{w}\}$.

**A Technical Remark About Goodness.** The notions of good and bad steps are inherently micro-step, as the next example shows. Consider the following term:

$$t := [f{\to}e]\langle\!|e?g|\!\rangle[o{\to}m]\langle\!|m\oslash e, n|\!\rangle n$$

In $t$, the cut $[f{\to}e]$ gives rise to *two* redexes, one for each occurrence of $e$. The redex concerning $\langle\!|e?g|\!\rangle$ is good while the one concerning $\langle\!|m\oslash e, n|\!\rangle$ is bad, because its position $[f{\to}e]\langle\!|e?g|\!\rangle[o{\to}m]\langle\!|m\oslash\langle\cdot\rangle, n|\!\rangle n$ is a bad context ($m$ dominates the hole, and there is a cut on $m$). Thus, being good/bad is not a property of the cut $[f{\to}e]$, that is, it is not a small-step property, because not all the micro-step redexes in which a cut is involved share the same character. Consider now:

$$t := [f{\to}e]\langle\!|e?m|\!\rangle\langle\!|m\oslash e, n|\!\rangle n$$

In $t$, again, there are two redexes on $e$, a good one and a bad one. The difference is that now it is the good occurrence of $e$ that turns the other occurrence into a bad one (by dominating $m$, which dominates the second occurrence).

**Basic Properties of Good Contexts.** The next three lemmas collect basic facts about good contexts that are used in the proof of the rest of the section.

**Lemma 11.3** (Good context decomposition). *Let $C\langle C'\rangle$ be a good context. Then $C$ and $C'$ are good contexts.*

**Lemma 11.4** (Bad cannot outer good). *Let $G\langle t\rangle = B\langle s\rangle$. Then $B \not\prec G$, that is, either $G \prec B$ or $G \parallel B$.*

*Proof.* By induction on $G$. Note that $B$ cannot be empty, otherwise it would be good. Cases:

- $G = \langle\cdot\rangle$. Then $t = B\langle s\rangle$ and so $B \not\prec G$.
- The abstraction, bang, par, and dereliction cases follow from the *i.h.*
- $G = (G'', u)$. If $B = (B', u)$ then it follows from the *i.h.* If $B = (r, B')$ then $B \not\prec G$.
- The subtraction case is as the tensor case.
- $G = [v{\to}x]G'$. If $B = [v{\to}x]B'$ it follows from the *i.h.* If $B = [B'{\to}x]u$ then $B \not\prec G$. □

**Lemma 11.5** (Double contexts and replacements). *Let $\mathbb{C}$ be a double context and $t$ and $s$ be two terms. Then:*

*(1) $\mathbb{C}\langle\cdot, t\rangle$ is as capturing as $\mathbb{C}\langle\cdot, s\rangle$;*
*(2) $\mathtt{dv}(\mathbb{C}\langle\cdot, t\rangle) = \mathtt{dv}(\mathbb{C}\langle\cdot, s\rangle)$;*
*(3) $\mathbb{C}\langle\cdot, t\rangle$ is good if and only if $\mathbb{C}\langle\cdot, s\rangle$ is good.*

Note that the last point of Lemma 11.5 could have be equivalently stated with respect to *being bad*.

*Proof.* By induction on $\mathbb{C}$. The base cases are straightforward, and the inductive cases follow from the *i.h.* In the inductive case $\mathbb{C} = [v{\to}x]\mathbb{C}'$, the third point uses the second one.                                                                                            $\square$

**Sub-Term Property.** We are ready for the key point of the paper, the sub-term property of the good strategy. The proof is based on a natural local invariant about *bad values*, which are the sub-terms at risk of being duplicated or erased.

**Definition 11.6** (Bad values)**.** Given $t = C\langle v \rangle$, $v$ is a *bad value* of $t$ if $C$ is a bad context.

The local invariant shows that one good step cannot create bad values. Since strict equality of sub-terms is not preserved by reduction (because of on-the-fly $\alpha$-renaming), the invariant concerns the *size* of bad values, which is what is important for cost analyses. The sub-term property follows by induction on the length of cut elimination sequences.

**Lemma 11.7.**
(1) *If $C\langle[v{\to}x]C'\rangle$ is good then $C\langle C'\rangle$ is good.*
(2) *If $C\langle C'\rangle$ is bad then $C\langle[v{\to}x]C'\rangle$ is bad.*

*Proof.* Point 1 is by induction on $C$. Since a context is either good or bad (Lemma 11.1), Point 2 is simply the contrapositive of Point 1.                                                      $\square$

**Proposition 11.8** (Local sub-term invariant)**.** *Let $t \to_{\mathsf{G}} s = B\langle v \rangle$. Then $t = B'\langle v' \rangle$ for some $B'$ and $v'$ such that $|v| = |v'|$.*

*Proof.* Cases of $t \to_{\mathsf{G}} s = B\langle v \rangle$:

- $\to_{\mathtt{axm_1}}$. We have $t = C\langle[v_{\mathtt{m}}{\to}m]M\langle\!\langle m \rangle\!\rangle\rangle \to_{\mathtt{Gaxm_1}} C\langle M\langle\!\langle v_{\mathtt{m}} \rangle\!\rangle\rangle = s = B\langle v \rangle$. Since $C\langle[v_{\mathtt{m}}{\to}m]M\rangle$ is good, $C\langle M \rangle$ is good by Lemma 11.7, and by Lemma 11.3 both $C$ and $M$ are good. By Lemma 11.4, $B \not\prec C$ and there are two cases:
  - $C \parallel B$. Then there is a double context $\mathbb{C}_{C,B}$ such that $\mathbb{C}_{C,B}\langle \cdot, v \rangle = C$ and $\mathbb{C}_{C,B}\langle M\langle\!\langle v_{\mathtt{m}} \rangle\!\rangle, \cdot \rangle = B$. By Lemma 11.5, the context $B' := \mathbb{C}_{C,B}\langle[v_{\mathtt{m}}{\to}m]M\langle\!\langle m \rangle\!\rangle, \cdot \rangle$ is bad. Note that $B'\langle v \rangle = \mathbb{C}_{C,B}\langle[v_{\mathtt{m}}{\to}m]M\langle\!\langle m \rangle\!\rangle, v \rangle = C\langle[v_{\mathtt{m}}{\to}m]M\langle\!\langle m \rangle\!\rangle\rangle = t$, and so $B'$ isolates $v$ in $t$.
  - $C \prec B$. By Lemma 11.4, $B \not\prec C\langle M \rangle$ and there are two sub-cases:
    * $C\langle M \rangle \parallel B$. Then $B = C\langle C' \rangle$ for some $C'$ such that $M \parallel C'$. There is a double context $\mathbb{C}_{M,C'}$ such that $\mathbb{C}_{M,C'}\langle \cdot, v \rangle = M$ and $\mathbb{C}_{M,C'}\langle v_{\mathtt{m}}, \cdot \rangle = C'$. Note that $C\langle\mathbb{C}_{M,C'}\rangle\langle v_{\mathtt{m}}, \cdot \rangle =_{L.9.3} C\langle\mathbb{C}_{M,C'}\langle v_{\mathtt{m}}, \cdot \rangle\rangle = C\langle C' \rangle = B$. By Lemma 11.5, the context $B' := C\langle\mathbb{C}_{M,C'}\rangle\langle m, \cdot \rangle$ is bad. Now, $B' =_{L.9.3} C\langle\mathbb{C}_{M,C'}\langle m, \cdot \rangle\rangle$ and let $C'' := \mathbb{C}_{M,C'}\langle m, \cdot \rangle$, so that $B' = C\langle C'' \rangle$. Note that $B'\langle v \rangle = C\langle\mathbb{C}_{M,C'}\langle m, v \rangle\rangle = C\langle M\langle m \rangle\rangle$ and also $B'\langle v \rangle = C\langle C''\langle v \rangle\rangle$. Then $t = C\langle[v_{\mathtt{m}}{\to}m]M\langle m \rangle\rangle = C\langle[v_{\mathtt{m}}{\to}m]C''\langle v \rangle\rangle$. Thus, $C\langle[v_{\mathtt{m}}{\to}m]C''\rangle$ isolates $v$ in $t$ and it is bad because of Lemma 11.7.2 and the fact that $B' = C\langle C'' \rangle$ is bad.
    * $C\langle M \rangle \prec B$. Then $B = C\langle M\langle C' \rangle\rangle$ for $C'$ such that $v_{\mathtt{m}} = C'\langle v \rangle$. The context $C\langle[C'{\to}m]M\langle\!\langle m \rangle\!\rangle\rangle$ is bad and isolates $v$ in $t$.
- $\to_{\mathtt{axm_2}}$. We have $t = C\langle[m{\to}n]r\rangle \to_{\mathtt{Gaxm_2}} C\langle\{m{\to}n\}r\rangle = s = B\langle v \rangle$. All bad values of $s$ coming from $C$ are obviously also in $t$. Let $B = C\langle\{m{\to}n\}C'\rangle$ be the position of a bad value $v$ in $s$. Two cases:
  - $C\langle C' \rangle$ *is a bad context.* Then $v$ occurs in $t$ as a bad value at position $C\langle[v_{\mathtt{m}}{\to}m]C\rangle$ (which is bad by Lemma 11.7.2).

- $C\langle C'\rangle$ *is a good context.* Since it is the addition of $\{m\!\to\!n\}$ that makes $B$ bad, necessarily $m \in \mathtt{dv}(C')$. Now, note that then $[v_\mathtt{m}\!\to\!m]C'$ is bad, and so is $C\langle[v_\mathtt{m}\!\to\!m]C'\rangle$, which isolates $v$ in $t$, making it a bad value.

- $\to_\otimes$. We have $t = C\langle[v_\mathtt{m}\!\to\!m]M\langle\!(m\,\mathfrak{V}x, y|r\rangle\rangle \to_{\mathtt{G}\otimes} C\langle M\langle L\langle[v'\!\to\!x]L'\langle[v''\!\to\!y]r\rangle\rangle\rangle\rangle = s$ with $v_\mathtt{m} = (L\langle v'\rangle, L'\langle v''\rangle)$ and $s = B\langle v\rangle$. If the hole of $B$ falls in $C$, $M$, or $L\langle[v'\!\to\!x]L'\langle[v''\!\to\!y]\langle\cdot\rangle\rangle\rangle$ then the reasoning is as in the $\to_{\mathtt{axm}_1}$ case. The only potentially delicate point is about the bad values in $r$. Let $B = C\langle M\langle L\langle[v'\!\to\!x]L'\langle[v''\!\to\!y]C'\rangle\rangle\rangle\rangle$ be the position of one such bad value $v$. Two cases:
  - $C\langle M\langle C'\rangle\rangle$ *is a bad context.* Then $v$ occurs in $t$ as bad value at position $C\langle[v_\mathtt{m}\!\to\!m]M\langle C'\rangle\rangle$.
  - $C\langle M\langle C'\rangle\rangle$ *is a good context.* Note that the only variables of $C'$ that can be captured by $L\langle[v'\!\to\!x]L'\langle[v''\!\to\!y]\langle\cdot\rangle\rangle\rangle$ are $x$ and $y$. Since it is the addition of $L\langle[v'\!\to\!x]L'\langle[v''\!\to\!y]\langle\cdot\rangle\rangle\rangle$ that makes $B$ bad, necessarily $x \in \mathtt{dv}(C')$ or $y \in \mathtt{dv}(C')$. Now, note that then $m \in \mathtt{dv}(M\langle\!(m\,\mathfrak{V}x, y|C'\rangle)$ and so that $C\langle[v_\mathtt{m}\!\to\!m]M\langle\!(m\,\mathfrak{V}x, y|C'\rangle\rangle$ is a bad context isolating $v$ in $t$.

- $\to_{\multimap}$. As for $\to_\otimes$.

- $\to_{\mathtt{axe}_1}$. We have $t = C\langle[v_\mathtt{e}\!\to\!e]C'\langle\!\langle e\rangle\!\rangle\rangle \to_{\mathtt{Gaxe}_1} C\langle[v_\mathtt{e}\!\to\!f]C'\langle\!\langle v_\mathtt{e}\rangle\!\rangle\rangle = s = B\langle v\rangle$. Since $C\langle[v_\mathtt{e}\!\to\!e]C'\rangle$ is good, by Lemma 11.4, $B \not\prec C\langle[v_\mathtt{e}\!\to\!e]C'\rangle$ and there are two cases:
  - $C\langle[v_\mathtt{e}\!\to\!e]C'\rangle \parallel B$. Then there is a double context $\mathbb{C}_{C\langle[v_\mathtt{e}\!\to\!e]C'\rangle,B}$ such that $\mathbb{C}_{C\langle[v_\mathtt{e}\!\to\!e]C'\rangle,B}\langle v_\mathtt{e}, \cdot\rangle = B$. By Lemma 11.5, the context $B' := \mathbb{C}_{C\langle[v_\mathtt{e}\!\to\!e]C'\rangle,B}\langle e, \cdot\rangle$ is bad. Note that $B'$ isolates $v$ in $t$.
  - $C\langle[v_\mathtt{e}\!\to\!e]C'\rangle \prec B$. Then $B = C\langle[v_\mathtt{e}\!\to\!e]C'\langle C''\rangle\rangle$ for $C''$ such that $v_\mathtt{e} = C''\langle v\rangle$. Note that the context $C\langle[C''\!\to\!e]C'\langle\!\langle e\rangle\!\rangle\rangle$ is bad and isolates $v$ in $t$.

- $\to_{\mathtt{axe}_2}$. We have $t = C\langle[f\!\to\!e]C'\langle\!(e?x|u\rangle\rangle \to_{\mathtt{Gaxe}_2} C\langle[f\!\to\!e]C'\langle\!(f?x|u\rangle\rangle = s = B\langle v\rangle$ with $C'$ not capturing $e$ nor $f$. If the hole of $B$ falls in $C\langle[f\!\to\!e]C'\rangle$ then the reasoning is as in the $\to_{\mathtt{axe}_1}$ case. The only potentially delicate point is about the bad values in $u$. Let $B = C\langle[f\!\to\!e]C'\langle\!(f?x|C''\rangle\rangle$ be the position of one such bad value $v$. The same value is isolated in $t$ by $B' = C\langle[f\!\to\!e]C'\langle\!(e?x|C''\rangle\rangle$. Note that $B'$ cannot be good: since the only thing that has changed is the renaming of $f$ as $e$ in the dereliction, then:
  - either $f$ is irrelevant for the badness of $B$, and so $B'$ is bad,
  - or $B$ is bad because $f \in \mathtt{dv}(\!(f?x|C'')$ and $f$ is cut in $C$. Note that then $e \in \mathtt{dv}(\!(e?x|C'')$ and $e$ is cut in $B'$, so that $B'$ is bad.

- $\to_!$. We have $t = C\langle[!u\!\to\!e]C'\langle\!(e?x|r\rangle\rangle \to_{\mathtt{G!}} C\langle[!u\!\to\!e]C'\langle L\langle[v'\!\to\!x]r\rangle\rangle\rangle = s = B\langle v\rangle$ with $u = L\langle v'\rangle$. If the hole of $B$ falls in $C\langle[!u\!\to\!e]C'\rangle$ then the reasoning is as in the $\to_{\mathtt{axe}_1}$ case. If it falls in $L\langle[v'\!\to\!x]\langle\cdot\rangle\rangle$, then a bad context with the hole in $u$ isolates the same value in $t$. The only potentially delicate point is about the bad values in $r$. Let $B = C\langle[!u\!\to\!e]C'\langle L\langle[v'\!\to\!x]C''\rangle\rangle\rangle$ be the position of one such bad value $v$. The same value is isolated in $t$ by $B' = C\langle[!u\!\to\!e]C'\langle\!(e?x|C''\rangle\rangle$ which cannot be good because:
  - either $x$ is irrelevant for the badness of $B$, and so $B'$ is bad because $x$ is the only variable of $C''$ that can be captured by $L\langle[v'\!\to\!x]\langle\cdot\rangle\rangle$ (which is the only difference between $B$ and $B'$);
  - or $B$ is bad because $x \in \mathtt{dv}(C'')$ and $x$ is cut in $B$. Note that then $e \in \mathtt{dv}(\!(e?x|C'')$ and $e$ is cut in $B'$, so that then $B'$ is also bad.

- $\to_\mathtt{w}$. We have $t = C\langle[v_\mathtt{e}\!\to\!e]r\rangle \to_{\mathtt{Gw}} C\langle r\rangle = s = B\langle v\rangle$ with $e \notin \mathtt{fv}(r)$. Since $C\langle[v_\mathtt{e}\!\to\!e]\langle\cdot\rangle\rangle$ is good, $C$ is good. By Lemma 11.4, $B \not\prec C$ and there are two cases:
  - $C \parallel B$. Then there is a double context $\mathbb{C}_{C,B}$ such that $\mathbb{C}_{C,B}\langle r, \langle\cdot\rangle\rangle = B$. By Lemma 11.5, the context $B' := \mathbb{C}_{C,B}\langle[v_\mathtt{e}\!\to\!e]r, \cdot\rangle$ is bad. Note that $B'$ isolates $v$ in $t$.

   – $C \prec B$. Then $B = C\langle C''\rangle$ for $C''$ such that $r = C''\langle v\rangle$. Note that the context $C\langle[v_{\mathsf{e}}{\to}e]C''\rangle$ is bad and isolates $v$ in $t$.      □

**Theorem 11.9** (Sub-term property). *Let $t \to_{\mathsf{G}}^* s$ and $v$ be a bad value of $s$. Then $|v| \leq |t|$.*

*Proof.* By induction on the length $k$ of the reduction $t \to_{\mathsf{G}}^k s$. If $k = 0$ then the statement trivially holds. If $k > 0$ then consider the last step $u \to_{\mathsf{G}} s$ of the sequence. By the local sub-term property (Prop. 11.8), the size of every bad value $v$ of $s$ is bound by the size of a bad value $v'$ of $u$, which by *i.h.* satisfy the statement.      □

**Good Diamond.** The good strategy is not deterministic, as for instance if $t \to_{\mathsf{G}} s$ then:
$$(s, t) \quad_{\mathsf{G}}{\leftarrow} \quad (t, t) \quad \to_{\mathsf{G}} \quad (t, s).$$
Note also, however, that the local diagrams that are not squares nor triangles are forbidden by design for good steps (see above). Triangles are also forbidden: it is easily seen that they all involve touching the cut value of a $\to_{\mathsf{w}}$ step, which is bad. Then all local diagrams for good steps are squares.

**Proposition 11.10** (Good diamond). *$s_1 {}_{\mathsf{G}}{\leftarrow} t \to_{\mathsf{G}} s_2$ and $s_1 \neq s_2$ then $s_1 \to_{\mathsf{G}} u {}_{\mathsf{G}}{\leftarrow} s_2$ for some $u$.*

*Proof.* For the proof to go through, we need to slightly strengthen the statement adding a clause about dominating free variables (to be used in the inductive cases). We also reformulate the statement using different meta variables. The new statement is:

    If $G_1 : t_0 \to_{\mathsf{G}} t_1$ and $G_2 : t_0 \to_{\mathsf{G}} t_2$ with $t_1 \neq t_2$ then that there exists $t_3$ such that
    $G_2' : t_1 \to_{\mathsf{G}} t_3$ with $\mathtt{dv}(G_2') = \mathtt{dv}(G_2)$ and $G_1' : t_2 \to_{\mathsf{G}} t_3$ with $\mathtt{dv}(G_1') = \mathtt{dv}(G_1)$.

    Let $a, b \in \{\mathtt{axm}_2, \mathtt{axm}_1, \otimes, {\multimap}, \mathtt{axe}_1, \mathtt{axe}_2, !, \mathtt{w}\}$, $t_0 \to_{\mathsf{G}a} t_1$, and $t_0 \to_{\mathsf{G}b} t_2$. The proof is by induction on $t_0 \to_{\mathsf{G}a} t_1$, that is, by induction on the context closing the root rule $\mapsto_a$, and case analysis of $t_0 \to_{\mathsf{G}b} t_2$. It is a check of diagrams similar to the proof of local confluence. As for local confluence, the deformation lemmas are used, but their statements have to be strengthened because they have to preserve goodness and dominating variables. About the diagrams, there are less cases, none of which is surprising. The details are in [Acc22].     □

    Two well known consequences of being diamond are *uniform normalization*, that is, if there is a normalizing reduction sequence then there are no diverging sequences, and *random descent*, that is, when a term is normalizable, all sequences to normal form *have the same length*. Being diamond is then a relaxed form of determinism. These properties are essential for the study of cost models, as otherwise the number of steps of the strategy is an ambiguously defined measure.

**Fullness.** The sub-term property expresses the correctness of our design. We need to prove a simple form of completeness, ensuring that the good strategy does not stop when there is still work left to do. This is given by *fullness*, that is, the fact that as long as $t$ is not normal then $t$ has a good redex. We need to assume that $t$ has no clashes, because terms with clashes such as $[([e{\to}f]f, e){\to}m](\!(m{\ominus}e, n)\!)n$, where the tensor pair and the subtraction clash, are not normal (note that $[e{\to}f]f$ is a redex) but have no good redexes, as all their redexes are inside cut values of clashing cuts.

    We recall that an *occurrence* of $x$ in $t$ is given by a position $t = C\langle t_x\rangle$ where $t_x$ is a sub-term of $t$ of shape $x$, $(\!(x \mathbin{\mathparagraph} y, z)\!)s$, $(\!(x{\oslash}v, y)\!)s$, or $(\!(x?y)\!)s$. Note that the shapes of $t_x$ are the kind of terms with which a cut on $x$ can interact at a distance to form a redex.

**Definition 11.11** (Outermost variable occurrence). An occurrence $t = C\langle t_x \rangle$ of $x$ in $t$ is *outermost* if $C' \not\prec C$ for all other occurrences $t = C'\langle t'_x \rangle$ of $x$.

**Lemma 11.12.** *Let $t = C\langle t_x \rangle$ be an outermost occurrence of $x$ in $t$ with $C$ not capturing $x$. Then $x \notin \mathtt{dv}(C)$.*

*Proof.* By induction on the number $|C|_x$ of free occurrences of $x$ in $C$. If $|C|_x = 0$ then $x \notin \mathtt{fv}(C) \supset \mathtt{dv}(C)$. If $|C|_x > 0$ consider another occurrence $C'\langle t'_x \rangle$ of $x$ in $t$ such that $C \not\prec C'$, which exists because $|C|_x > 0$ (indeed if $C \prec C'$ then $t'_x$ is a sub-term of $t_x$ and does not contribute to $|C|_x$). Since $t_x$ is outermost, we have $C' \parallel C$. Now, consider the double context $\mathbb{C}_{C,C'}$ such that $\mathbb{C}_{C,C'}\langle \cdot, t'_x \rangle = C$ and $\mathbb{C}_{C,C'}\langle t_x, \cdot \rangle = C'$. By Lemma 11.5, the context $C'' := \mathbb{C}_{C,C'}\langle \cdot, \lambda m.m \rangle$ verifies $\mathtt{dv}(C'') = \mathtt{dv}(C)$ and $|C''|_x \leq |C|_x - 1$. Then by *i.h.* we have $x \notin \mathtt{dv}(C'') = \mathtt{dv}(C)$. $\qquad\square$

**Lemma 11.13.** *Let $x \in \mathtt{dv}(C)$. Then one of the following cases hold:*

(1) $C = M\langle (\!| m \invamp y, z |\!) C' \rangle$ *and* $x = m$;
(2) $C = M\langle (\!| m \otimes v, y |\!) C' \rangle$ *and* $x = m$;
(3) $C = M\langle (\!| m \otimes V, y |\!) t \rangle$ *and* $x = m$;
(4) $C = C'\langle (\!| e?y |\!) C'' \rangle$, $x = e$, *and* $e \notin \mathtt{dv}(C')$.

*Proof.* The first three points are simply the observation that if $x = m$ is multiplicative and $m \in \mathtt{dv}(C)$ then

- $m \in \mathtt{fv}(C)$ and so the position of $m$ has to be a multiplicative context $M$;
- the occurrence has to be a par or subtraction occurrence by definition of $\mathtt{dv}(C)$ (that is, it cannot be a variable occurrence).

The fourth point is similar but it says something more so we need a bit more. It is by induction on $C$. If $x = e$ is exponential and $e \in \mathtt{dv}(C)$ then $C = C''\langle (\!| e?y |\!) C''' \rangle$. Now, if $e \notin \mathtt{dv}(C'')$ the statement holds with $C' := C''$. Otherwise, by *i.h.*, $C'' = C'''\langle (\!| e?z |\!) C'''' \rangle$ with $e \notin \mathtt{dv}(C''')$. Then $C = C'''\langle (\!| e?z |\!) C''''\langle (\!| e?y |\!) C''' \rangle \rangle$, which satisfies the statement with respect to $C' := C'''$. $\qquad\square$

**Proposition 11.14** (Fullness). *Let $t$ be clash-free. If $t$ is not $\rightarrow_{\mathtt{ms}}$-normal then $t \rightarrow_{\mathtt{G}} s$ for some $s$.*

*Proof.* By induction on $t$. For variables the statement trivially holds, because they are $\rightarrow_{\mathtt{ms}}$-normal. All inductive cases but cut immediately follow from the *i.h.* Then let $t = [v \rightarrow x]u$. Two cases:

(1) $x \notin \mathtt{fv}(u)$. Then $t = [v \rightarrow x]u \rightarrow_{\mathtt{w}} u$. Note that this step is good, so the statement holds with $s := u$.
(2) $x \in \mathtt{fv}(u)$. Two sub-cases:
    (a) $u$ *is* $\rightarrow_{\mathtt{ms}}$-*normal.* Let $u = C\langle u_x \rangle$ with $u_x$ an outermost occurrence of $x$ in $u$ (clearly $C$ does not capture $x$). We show that the context $[v \rightarrow x]C$ is good. By Lemma 7.6, $u$ is cut-free (because $t$ is clash-free and $\rightarrow_{\mathtt{ms}}$-normal). Since there are no cuts in $C$, $C$ is good. By Lemma 11.12, $x \notin \mathtt{dv}(C)$. Then $[v \rightarrow x]C$ is good. By clash-freeness, $[v \rightarrow x]C$ is the position of a redex, which is then good.
    (b) $u$ *is not* $\rightarrow_{\mathtt{ms}}$-*normal.* By *i.h.*, $u$ has a good redex. Let its position be $G$, with $u = G\langle r \rangle$. If $[v \rightarrow x]G$ is a good context then it induces the redex for the statement. If instead it is a bad context then it must be that $x \in \mathtt{dv}(G)$. By Lemma 11.13, $G$ can have 4 possible shapes:

(i) *x has a par occurrence*: $G = M\langle\!\langle m \,⅋\, y, z\rangle\!\rangle C\rangle$ and $x = m$. By Lemma 11.3 and the fact that $G$ is good, $M$ is good. Then $[v{\to}m]M$ is good, because (being a multiplicative variable) $m \notin \mathtt{fv}(M) \supseteq \mathtt{dv}(M)$. Since $t$ has no clashes, $v$ is a tensor pair and $[v{\to}m]M$ is the position of a good redex.

(ii) *x has a subtraction occurrence 1*: $G = M\langle\!\langle m\oslash v, y\rangle\!\rangle C\rangle$ and $x = m$. Analogously to the par occurrence.

(iii) *x has a subtraction occurrence 2*: $G = M\langle\!\langle m\oslash C, y\rangle\!\rangle t\rangle$ and $x = m$. Analogously to the par occurrence.

(iv) *x has a dereliction occurrence*: $G = C\langle\!\langle e?C'\rangle\!\rangle\rangle$, $x = e$, and $e \notin \mathtt{dv}(C)$. This case is also similar to the par occurrence, the only difference is that $e \notin \mathtt{dv}(C)$ is given by Lemma 11.13.4 instead that obtained by linearity as in the multiplicative cases.                                                                              $\square$

**Summing Up.** Together with typed strong normalization (from Sect. 13), the properties of the good strategy imply our main result.

**Theorem 11.15** (Good polynomial cost model)**.** *Let $\Gamma \vdash t : A$ be a typed term. Then there exist $k$ and a cut-free term $s$ such that $t \to_{\mathsf{G}}^k s$. Moreover, such a reduction sequence is implementable on a random access machine in time polynomial in $k$ and $|t|$.*

*Proof.* By typed strong normalization (Corollary 13.13), $\to_{\mathsf{G}}$ terminates on $t$, let $s$ be its $\to_{\mathsf{G}}$ normal form. Since typable implies clash-freeness (Lemma 7.6), $t$ is clash-free, and thus so is $s$. By fullness (Prop. 11.14) and $\to_{\mathsf{G}}$ normality, $s$ is $\to_{\mathsf{ms}}$-normal, which with clash-freeness implies that it is cut-free (Lemma 7.6).

Now, about the cost. Let $t = t_0 \to_{\mathsf{G}} t_1 \to_{\mathsf{G}} \ldots \to_{\mathsf{G}} t_k = s$. All steps $t_i \to_{\mathsf{G}} t_{i+1}$ of the sequence that are multiplicative, as well as all $\to_{\mathtt{axe}_2}$ step, do only manipulation of a constant number of constructors of the term. Therefore, they can clearly be implemented in time polynomial in $|t_i|$, as searching for a redex and checking that its position is good are clearly polynomial problems, and there is at most a linear number of redexes in a term. By a straightforward induction using the sub-term property (Theorem 11.9), one obtains that $|t_i| \leq i \cdot |t_0|$, and so all those steps have cost polynomial in $k$ and $|t_0|$. Again by the sub-term property, the cost of duplication/erasure of values in non-linear steps (namely, $\to_{\mathtt{axe}_1}$, $\to_{!}$, and $\to_{\mathtt{w}}$ steps) is bound by $t_0$, so they have polynomial cost in $k$ and $|t_0|$ as well.      $\square$

About the degree of the polynomial, Accattoli et al. [ABM14, ABM15] show that strategies with the sub-term property are usually implementable via abstract machines in time *linear* in both $k$ and $|t|$. We expect the linear bound to hold here as well, but we leave the design of an abstract machine to future work.

## 12. Untyped Preservation of Strong Normalization

In the ES literature, the crucial property for a calculus with ES is *preservation of $\beta$ strong normalization* (shortened to PSN) on untyped terms, that is, that if a $\lambda$-term is $SN_\beta$ then it is SN when evaluated with ES. In our setting, it is rephrased as $t \in SN_{\to_{\mathsf{ss}}}$ implies $t \in SN_{\to_{\mathsf{ms}}}$. The property ensures that switching to micro-steps does not introduces unexpected divergence, as it was surprisingly the case, historically, for the first $\lambda$-calculus with ES by Abadi et al. [ACCL91], as shown by Melliès [Mel95].

Proofs of PSN are often demanding, see Kesner's survey [Kes07]. Contextual rewriting rules enable instead short proofs, as first showed by Accattoli and Kesner [AK10]. We adapt and further simplify that proof, that here rests on just two natural properties of micro steps, here referred to as *extension* and *root cut expansion* (Kesner calls the latter *IE property*), the latter making use of *structural stability*, that is, the fact that cut equivalence preserves SN (Prop. 7.11.2). The properties are actually also used for strong normalization in the next section, where they are used for $\to_{\mathtt{ss}}$, which is why the statements concern both $\to_{\mathtt{ms}}$ and $\to_{\mathtt{ss}}$.

**Extension.** The *extension* property is the easy fact that, for all root constructs but (non clashing) cuts, SN follows from SN of the root sub-terms.

**Lemma 12.1** (Extension). *Let $\to \in \{\to_{\mathtt{ss}}, \to_{\mathtt{ms}}\}$ and $t, s, v \in SN_\to$. Then:*
(1) Neutral*: $(t, s)$, $\lambda x.t$, $!t$, $(\!|m\,\Im x, y|\!)t$, $(\!|m\otimes v, x|\!)t$, and $(\!|e?x|\!)t$ are in $SN_\to$;*
(2) Clash*: if the root cut of $[v{\to}x]t$ is clashing then $[v{\to}x]t \in SN_\to$.*

*Proof.* By induction on (in the cases with more than one induction the order between the components is irrelevant):
- $t \in SN_\to$ for $\lambda x.t$, $!t$, $(\!|m\,\Im x, y|\!)t$, and $(\!|e?x|\!)t$,
- $(t \in SN_\to, s \in SN_\to)$ for $(t, s)$,
- $(t \in SN_\to, v \in SN_\to)$ for $(\!|m\otimes v, x|\!)t$,
- $(t \in SN_\to, v_{\mathtt{m}} \in SN_\to)$ for $[v_{\mathtt{m}}{\to}e]t$,
- $(t \in SN_\to, v_{\mathtt{e}} \in SN_\to)$ for $[v_{\mathtt{e}}{\to}m]t$.

In each case one shows that all reducts are in $SN_\to$, which follows immediately from the *i.h.*, because there cannot be interaction between the immediate sub-terms. $\qquad\square$

**Root Cut Expansion.** The *root cut expansion* property is the less obvious fact that if the reduct of a root *small-step* cut is $SN_{\to_{\mathtt{ms}}}$ then the reducing term also is. Kesner isolated the importance of this property for PSN [Kes09], calling it *IE property*. Accattoli and Kesner [AK10, Acc13] show that, with rules *at a distance*, it is proved via simple inductions (here even simpler than in [AK10, Acc13] thanks to LSC-style duplication and structural stability), while commutative rules (such as Girard's box commutation $\to_{\boxdot}$ for proof nets) break those inductions and require more complex techniques.

**Proposition 12.2** (Root cut expansion). *Let $\to \in \{\to_{\mathtt{ss}}, \to_{\mathtt{ms}}\}$*
(1) Multiplicative*: if $t \mapsto_{\mathtt{m}} s$ and $s \in SN_\to$ then $t \in SN_\to$.*
(2) Exponential*: if $v_{\mathtt{e}} \in SN_\to$ and $\{v_{\mathtt{e}}{\to}e\}t \in SN_\to$ then $[v_{\mathtt{e}}{\to}e]t \in SN_\to$.*

*Proof.*
(1) By induction on $s \in SN_\to$, showing that any reduct of $t$ is in $SN_\to$. Cases of the multiplicative root step:
- $\mapsto_{\mathtt{axm_1}}$, that is, $t = [v_{\mathtt{m}}{\to}m]M\langle\!\langle m\rangle\!\rangle \mapsto_{\mathtt{axm_1}} M\langle\!\langle v_{\mathtt{m}}\rangle\!\rangle = s$. If it is the root cut that it is reduced then $s \in SN_\to$ by hypothesis. The other cases of reduction in $t$ are:
  - *Reduction of a cut of $M\langle\!\langle m\rangle\!\rangle$*: that is, $[v_{\mathtt{m}}{\to}m]M\langle\!\langle m\rangle\!\rangle \to [v_{\mathtt{m}}{\to}m]u$ because $M\langle\!\langle m\rangle\!\rangle \to u$. By deformation (Lemma 9.4.1), there are two cases:
    (a) $u = M'\langle\!\langle m\rangle\!\rangle$ and $M\langle\!\langle v_{\mathtt{m}}\rangle\!\rangle \to M'\langle\!\langle v_{\mathtt{m}}\rangle\!\rangle$. Then we have the following diagram:

$$
\begin{array}{ccccccc}
t & = & [v_{\mathtt{m}}{\to}m]M\langle\!\langle m\rangle\!\rangle & \mapsto_{\mathtt{axm_1}} & M\langle\!\langle v_{\mathtt{m}}\rangle\!\rangle & = & s \\
 & & \downarrow & & \downarrow & & \\
 & & [v_{\mathtt{m}}{\to}m]M'\langle\!\langle m\rangle\!\rangle & \mapsto_{\mathtt{axm_2}} & M'\langle\!\langle v_{\mathtt{m}}\rangle\!\rangle & &
\end{array}
$$

By *i.h.* applied to the bottom side of the diagram, $[v_\mathtt{m}{\to}m]M'\langle\!\langle m\rangle\!\rangle \in SN_\to$.

(b) $M = M'\langle[\langle\cdot\rangle{\to}n]u'\rangle$ for some $M'$ and the step $M\langle\!\langle m\rangle\!\rangle \to u$ reduces the cut on $n$, that is, $u' = M''\langle u_n\rangle$ for some $M''$ and variable occurrence $u_n$, and $M\langle\!\langle m\rangle\!\rangle = M'\langle[m{\to}n]M''\langle u_n\rangle\rangle \to_{\mathtt{axm}_2} M'\langle M''\langle\{m{\to}n\}u_n\rangle\rangle = u$. We have:

$$t \;=\; [v_\mathtt{m}{\to}m]M'\langle[m{\to}n]M''\langle u_n\rangle\rangle \;\mapsto_{\mathtt{axm}_1}\; M'\langle[v_\mathtt{m}{\to}n]M''\langle u_n\rangle\rangle \;=\; s$$
$$\downarrow$$
$$t' \;=\; [v_\mathtt{m}{\to}m]M'\langle M''\langle\{m{\to}n\}u_n\rangle\rangle$$

Note that $t' \equiv_{\mathtt{cut}} s$. By the fact that cut equivalence preserves SN (Prop. 7.11) and $s \in SN_\to$, we obtain $t' \in SN_\to$.

– *Reduction of a cut of $v_\mathtt{m}$*: then $[v_\mathtt{m}{\to}m]M\langle\!\langle m\rangle\!\rangle \to [v'_\mathtt{m}{\to}m]M\langle\!\langle m\rangle\!\rangle$ with $v_\mathtt{m} \to v'_\mathtt{m}$. We have the following diagram:

$$t \;=\; [v_\mathtt{m}{\to}m]M\langle\!\langle m\rangle\!\rangle \;\mapsto_{\mathtt{axm}_1}\; M\langle\!\langle v_\mathtt{m}\rangle\!\rangle \;=\; s$$
$$\downarrow \qquad\qquad\qquad \downarrow$$
$$[v'_\mathtt{m}{\to}m]M\langle\!\langle m\rangle\!\rangle \;\mapsto_{\mathtt{axm}_1}\; M\langle\!\langle v'_\mathtt{m}\rangle\!\rangle$$

By *i.h.* applied to the bottom side of the diagram, we obtain $[v'_\mathtt{m}{\to}m]M\langle\!\langle m\rangle\!\rangle \in SN_\to$.

- $\mapsto_{\mathtt{axm}_2}$, that is, $t = [y{\to}x]u \mapsto_{\mathtt{axm}_2} \{y{\to}x\}u = s$. If it is the root cut that it is reduced then its reduct is in $SN_\to$ by hypothesis. The other cases of reduction in $t$ are:

- *Reduction of a cut of $u$*, that is $[y{\to}x]u \to [y{\to}x]r$ because $u \to_\mathtt{m} r$. We have:

$$t \;=\; [y{\to}x]u \;\mapsto_{\mathtt{axm}_2}\; \{y{\to}x\}u \;=\; s$$
$$\downarrow \qquad\qquad \downarrow$$
$$[y{\to}x]r \;\mapsto_{\mathtt{axm}_2}\; \{y{\to}x\}r$$

where the step on the right side is given by the stability of steps under renaming (Lemma 7.12.1). By *i.h.* applied to the bottom side of the diagram, we obtain $[y{\to}x]r \in SN_\to$.

- $\mapsto_\otimes$, that is, $t = [v_\mathtt{m}{\to}m]M\langle\!(m\mathbin{⅋}x, y)\!|p\rangle \mapsto_\otimes M\langle L\langle[v{\to}x]L'\langle[v'{\to}y]p\rangle\rangle\rangle$ with $v_\mathtt{m} = (u, r) = (L\langle v\rangle, L'\langle v'\rangle)$. If it is the root cut that it is reduced then its reduct is in $SN_\to$ by hypothesis. The other cases of reduction in $t$ are:

- *Reduction of a cut of $M\langle\!(m\mathbin{⅋}x, y)\!|p\rangle$*: then $t = [v_\mathtt{m}{\to}m]M\langle\!(m\mathbin{⅋}x, y)\!|p\rangle \to_\mathtt{ms} [v_\mathtt{m}{\to}m]t'$ because $M\langle\!(m\mathbin{⅋}x, y)\!|p\rangle \to_\mathtt{ms} t'$. By deformation (Lemma 9.4.3), $t' = M'\langle\!(m\mathbin{⅋}x, y)\!|p'\rangle$ for some $M'$ and $p'$ and also $M\langle L\langle[v{\to}x]L'\langle[v'{\to}y]p\rangle\rangle\rangle \to_\mathtt{ms} M'\langle L\langle[v{\to}x]L'\langle[v'{\to}y]p'\rangle\rangle\rangle$. Then we have the following diagram:

$$t \;=\; [v_\mathtt{m}{\to}m]M\langle\!(m\mathbin{⅋}x, y)\!|p\rangle \;\mapsto_\otimes\; M\langle L\langle[v{\to}x]L'\langle[v'{\to}y]p\rangle\rangle\rangle \;=\; s$$
$$\downarrow \qquad\qquad\qquad\qquad \downarrow$$
$$[v_\mathtt{m}{\to}m]M'\langle\!(m\mathbin{⅋}x, y)\!|p'\rangle \;\mapsto_\otimes\; M'\langle L\langle[v{\to}x]L'\langle[v'{\to}y]p'\rangle\rangle\rangle$$

By *i.h.* applied to the bottom side of the diagram, we obtain $[v'_\mathtt{m}{\to}m]M\langle\!\langle m\rangle\!\rangle \in SN_\to$.

- *Reduction of a cut of $v_\mathtt{m}$*: if the redex affects only $v$, $v'$, $L$, or $L'$, then the obvious diagram together with the *i.h.* gives the statement. If the redex is given by a cut in $L$ acting on a variable occurrence in $v$, that is, $v_\mathtt{m} = (L\langle v\rangle, L'\langle v'\rangle) \to_\mathtt{ms} (L''\langle v''\rangle, L'\langle v'\rangle) = v'_\mathtt{m}$ then we have $L\langle[v{\to}x]L'\langle[v'{\to}y]p'\rangle\rangle \to_\mathtt{ms} L''\langle[v''{\to}x]L'\langle[v'{\to}y]p'\rangle\rangle$. Then we have the following diagram:

$$t = [v_\mathtt{m}{\to}m]M\langle\!(m\mathbin{⅋}x, y)\!|p\rangle \qquad\qquad \mapsto_\otimes \quad M\langle L\langle[v{\to}x]L'\langle[v'{\to}y]p\rangle\rangle\rangle = s$$
$$\downarrow \qquad\qquad\qquad\qquad \downarrow$$
$$[(L''\langle v''\rangle, L'\langle v'\rangle){\to}m]M'\langle\!(m\mathbin{⅋}x, y)\!|p'\rangle \quad \mapsto_\otimes \quad L''\langle[v''{\to}x]L'\langle[v'{\to}y]p'\rangle\rangle$$

By *i.h.* applied to the bottom side of the diagram, we obtain $[(L''\langle v''\rangle, L'\langle v'\rangle){\to}m]M'\langle\!(m\mathbin{⅋}x, y)\!|p'\rangle \in SN_\to$.

- $\mapsto_{\multimap}$: as the previous case, simply using Lemma 9.4.4 instead of Lemma 9.4.3 at the corresponding point of the proof.

(2) We first treat the case of $\to_{\mathtt{ss}}$. By induction on $(v_{\mathtt{e}} \in SN_{\to_{\mathtt{ss}}}, \{v_{\mathtt{e}}\to e\}t \in SN_{\to_{\mathtt{ss}}})$, proving that any reduct of $[v_{\mathtt{e}}\to e]t$ is in $SN_{\to_{\mathtt{ss}}}$. Cases:

- *Reduction of the root cut*: $[v_{\mathtt{e}}\to e]t \mapsto_! \{v_{\mathtt{e}}\to e\}t$, which is in $SN_{\to_{\mathtt{ss}}}$ by hypothesis.
- *Reduction of a cut of $t$*: that is, $[v_{\mathtt{e}}\to e]t \to_{\mathtt{ss}} [v_{\mathtt{e}}\to e]t'$ with $t \to_{\mathtt{ss}} t'$. By stability under substitution (Lemma 6.6.1), $\{v_{\mathtt{e}}\to e\}t \to_{\mathtt{ss}} \{v_{\mathtt{e}}\to e\}t'$. By *i.h.* (2nd component, the 1st is unchanged), $[v_{\mathtt{e}}\to e]t' \in SN_{\to_{\mathtt{ss}}}$.
- *Reduction of a cut of $v_{\mathtt{e}}$*: that is, $[v_{\mathtt{e}}\to e]t \to_{\mathtt{ss}} [v'_{\mathtt{e}}\to e]t$ with $v_{\mathtt{e}} \to_{\mathtt{ss}} v'_{\mathtt{e}}$. By Lemma 6.6.1, $\{v_{\mathtt{e}}\to e\}t \to^*_{\mathtt{ss}} \{v'_{\mathtt{e}}\to e\}t$, and so $\{v'_{\mathtt{e}}\to e\}t \in SN_{\to_{\mathtt{ss}}}$. By *i.h.* (1st component), $[v'_{\mathtt{e}}\to e]t \in SN_{\to_{\mathtt{ss}}}$.

Now, the case of $\to_{\mathtt{ms}}$, which is similar. By induction on $(v_{\mathtt{e}} \in SN_{\to_{\mathtt{ms}}}, \{v_{\mathtt{e}}\to e\}t \in SN_{\to_{\mathtt{ms}}}, |t|_e)$, proving that any reduct of $[v_{\mathtt{e}}\to e]t$ is in $SN_{\to_{\mathtt{ms}}}$. The cases where reduction takes place in $t$ or $v_{\mathtt{e}}$ go exactly as for $\to_{\mathtt{ss}}$. What changes is when the root cut is reduced. Cases:

- $\mapsto_{\mathtt{axe}_1}$, that is, $[v_{\mathtt{e}}\to e]t = [v_{\mathtt{e}}\to e]C\langle\!\langle e\rangle\!\rangle \mapsto_{\mathtt{axe}_1} [v_{\mathtt{e}}\to e]C\langle\!\langle v_{\mathtt{e}}\rangle\!\rangle$. Note that we can assume that $e \notin \mathtt{fv}(v_{\mathtt{e}})$. Since $\{v_{\mathtt{e}}\to e\}C\langle\!\langle e\rangle\!\rangle = \{v_{\mathtt{e}}\to e\}C\langle\!\langle v_{\mathtt{e}}\rangle\!\rangle$ and $|C\langle\!\langle v_{\mathtt{e}}\rangle\!\rangle|_e < |C\langle\!\langle e\rangle\!\rangle|_e$, we can apply the *i.h.* (3rd component, 1st and 2nd unchanged), obtaining $[v_{\mathtt{e}}\to e]C\langle\!\langle v_{\mathtt{e}}\rangle\!\rangle \in SN_{\to_{\mathtt{ms}}}$.
- $\mapsto_{\mathtt{axe}_2}$, that is, $v_{\mathtt{e}} = f$ and the step is $[f\to e]t = [f\to e]C\langle(\![e?x]\!)u\rangle \mapsto_{\mathtt{axe}_2} [f\to e]C\langle(\![f?x]\!)u\rangle$. Since $\{f\to e\}C\langle(\![e?x]\!)u\rangle = \{f\to e\}C\langle(\![f?x]\!)u\rangle$ and $|C\langle(\![f?x]\!)u\rangle|_e < |C\langle(\![e?x]\!)u\rangle|_e$, we can apply the *i.h.* (3rd component, 1st and 2nd unchanged), obtaining $[f\to e]C\langle(\![f?x]\!)u\rangle \in SN_{\to_{\mathtt{ms}}}$.
- $\mapsto_!$, that is, $v_{\mathtt{e}} = !s = !L\langle v\rangle$ and the step is:
$$[!s\to e]t = [!s\to e]C\langle(\![e?x]\!)u\rangle \mapsto_! [!s\to e]C\langle L\langle[v\to x]u\rangle\rangle$$
Note that we can assume $e \notin \mathtt{fv}(v_{\mathtt{e}})$. Since $\{!s\to e\}C\langle(\![e?x]\!)u\rangle = \{!s\to e\}C\langle L\langle[v\to x]u\rangle\rangle$ and $|C\langle L\langle[v\to x]u\rangle\rangle|_e < |C\langle(\![e?x]\!)u\rangle|_e$, we can apply the *i.h.* (3rd component, 1st and 2nd unchanged), obtaining $[!s\to e]C\langle L\langle[v\to x]u\rangle\rangle \in SN_{\to_{\mathtt{ms}}}$.
- $\mapsto_{\mathtt{w}}$, that is, $e \notin \mathtt{fv}(t)$ and $[v_{\mathtt{e}}\to e]t \mapsto_{\mathtt{w}} t$. Note that $t = \{v_{\mathtt{e}}\to e\}t$, which is in $SN_{\to_{\mathtt{ms}}}$ by hypothesis. $\qquad\square$

**PSN.** Now, PSN is proved via a very easy induction over $t \in SN_{\to_{\mathtt{ss}}}$ and the size $|t|$ of $t$, using the two properties.

**Theorem 12.3** (Untyped PSN). *If $t \in SN_{\to_{\mathtt{ss}}}$ then $t \in SN_{\to_{\mathtt{ms}}}$.*

*Proof.* By induction on $(t \in SN_{\to_{\mathtt{ss}}}, |t|)$. Cases of $t$:

- *All cases but cut*: it follows from the *i.h.* and neutral extension (Lemma 12.1). We spell out one case. Let $t = (\![m\oslash v, x]\!)s$. By *i.h.* (2nd component), $v, s \in SN_{\to_{\mathtt{ms}}}$. By extension, $t \in SN_{\to_{\mathtt{ms}}}$.
- *Root cut*, that is, $t = [v\to x]s$. There are 3 cases, handled by either clashing extension (Lemma 12.1) or root cut expansion (Prop. 12.2):
  (1) *The root cut is clashing*. By *i.h.* (2nd component), $v, s \in SN_{\to_{\mathtt{ms}}}$. By clashing extension, $t \in SN_{\to_{\mathtt{ms}}}$.
  (2) *The root cut is multiplicative and not clashing*. Then $t \mapsto_{\mathtt{m}} u$. By *i.h.* (1st component), $u \in SN_{\to_{\mathtt{ms}}}$. By multiplicative root cut expansion, $t \in SN_{\to_{\mathtt{ms}}}$.
  (3) *The root cut is exponential*. By *i.h.* (2nd component), $v \in SN_{\to_{\mathtt{ms}}}$. We have $t = [v_{\mathtt{e}}\to e]s \to_! \{v_{\mathtt{e}}\to e\}s$. By *i.h.* (1st component), $\{v_{\mathtt{e}}\to e\}s \in SN_{\to_{\mathtt{ms}}}$. By exponential root cut expansion, $[v_{\mathtt{e}}\to e]s \in SN_{\to_{\mathtt{ms}}}$. $\qquad\square$

The next section proves typed SN for $\rightarrow_{\text{ss}}$, which then transfers to $\rightarrow_{\text{ms}}$ by PSN. The relevance of PSN, however, is that it holds in the *untyped* setting, thus it transfers SN also if proved with respect to other type systems, such as intersection types, polymorphic types, and so on.

## 13. Typed Strong Normalization

Here we prove strong normalization of the IMELL-typed small-step ESC (then transferred to the micro-step ESC by PSN) using the reducibility method. We do it following the *bi-orthogonal* schema for proof nets by Girard [Gir87], see also Pagani and Tortora de Falco [PTdF10]. We actually adopt here the variant by Accattoli [Acc13], the main point of which is that without commutative exponential rules (as it is also the case here) there is a simpler proof of *adequacy* (the key step of the method) based on the *root cut expansion property* of the previous section. There are 3 differences between the present proof and [Acc13] (beyond a neater technical development here):

(1) *Formal*: the study in [Acc13] uses proof nets and an *informal* term notation with explicit and meta-level substitutions. That informal notation is here replaced by a formal one, the ESC;
(2) *Exponential rules*: both studies use rules without commutative cases, but here we use LSC-style duplications, obtaining a simpler proof of root cut expansion.
(3) *Quotient vs cut equivalence*: the quotient on proofs given by proof nets plays a subtle role in the proof of adequacy. The argument is here replaced by the stability by cut equivalence $\equiv_{\text{cut}}$ of strong normalization.

We are also inspired by Riba's dissection of reducibility [Rib08], and refer to [Acc13] for extensive discussions and references about strong normalization in linear logic. To our knowledge, the literature contains only two studies of strong normalization for a linear term calculus:

- Benton [Ben95], who, as he points out, does not deal with commutative cases—that are nonetheless crucial in his presentation—thus not really proving strong normalization;
- Pérez et al. [PCPT14], where cut elimination is seriously restricted because they use it to model process communication, which does not take place under prefixes. Logically, it means that their notion of cut elimination does not compute cut-free proofs.

**Key Properties.** The reducibility method requires a number of definitions. From a high-level perspective, however, our proof is based on three key properties of $\rightarrow_{\text{ss}}$:

(1) *Structural stability*: if $t \in SN_{\rightarrow_{\text{ss}}}$ and $t \equiv_{\text{cut}} s$ then $s \in SN_{\rightarrow_{\text{ss}}}$ (Prop. 7.11);
(2) *Extension* (Lemma 12.1), and
(3) *Root cut expansion* (Prop. 12.2).

Since they hold also for micro-step cut elimination $\rightarrow_{\text{ms}}$, our proof works with only minor changes also for micro-steps, without really needing PSN.

**Elimination Contexts and Duality.** Elimination contexts are multiplicative contexts of the form $L\langle[\langle\cdot\rangle\!\rightarrow\!x]t\rangle$, noted $E$. They can be typed by considering $\langle\cdot\rangle$ as a free variable and typing it via an axiom, as follows:

$$\frac{}{\langle\cdot\rangle : A \vdash \langle\cdot\rangle : A}\ \text{ax}_{\text{ctx}}$$

Let us set some notations:

- $\mathtt{Terms}_{\vdash A}$ for the set of terms $t$ of type $A$, that is, such that $\Gamma \vdash t : A$, and also $\mathcal{V}_A := \mathtt{Terms}_{\vdash A} \cap \mathcal{V}$.
- $\mathtt{ElCtxs}_A := \{E \mid \Gamma, \langle \cdot \rangle : A \vdash E : B\}$ the set of typed elimination contexts with hole of type $A$, and say that $E$ has *co-type* $A$.
- $\mathtt{ElVars}_A := \{E \in \mathtt{ElCtxs}_A \mid E = [\langle \cdot \rangle \to x] x\}$.
- $E \in SN_{\to_{\mathrm{ss}}}$ if $E = L\langle [\langle \cdot \rangle \to x] t \rangle$ and $L\langle \{y \to x\} t \rangle \in SN_{\to_{\mathrm{ss}}}$ for every variable $y$ of the same kind as $x$ which is not captured by $L$.

**Remark 13.1.** Checking that $E = L\langle [\langle \cdot \rangle \to x] t \rangle$ is in $SN_{\to_{\mathrm{ss}}}$ amounts to prove that $L\langle \{y \to x\} t \rangle = \{y \to x\} L\langle t \rangle \in SN_{\to_{\mathrm{ss}}}$ for every appropriate $y$. By the stability of SN by renamings (Lemma 7.12.2), it is enough to prove that $L\langle t \rangle \in SN_{\to_{\mathrm{ss}}}$.

**Definition 13.2** (Duality). Given a set $\mathcal{S} \subseteq \mathtt{Terms}_{\vdash A}$ of terms of type $A$, the *dual* set $\mathcal{S}^{\perp} \subseteq \mathtt{ElCtxs}_A$ contains the elimination contexts $E$ of co-type $A$ such that $E\langle\!\langle t \rangle\!\rangle$ is proper and in $SN_{\to_{\mathrm{ss}}}$ for every $t \in \mathcal{S}$. The dual of a set of elimination contexts $\mathcal{E} \subseteq \mathtt{ElCtxs}_A$ of co-type $A$ is defined analogously.

Note the use of $\langle\!\langle \cdot \rangle\!\rangle$ in the definition of duality: the plugging in contexts at work in duality does *not* capture the variables of the plugged term, this is crucial, and standard in the reducibility method.

The following properties of duality are standard.

**Lemma 13.3** (Basic properties of duality). *If $\mathcal{S} \subseteq \mathtt{Terms}_{\vdash A}$ or $\mathcal{S} \subseteq \mathtt{ElCtxs}_A$ then:*

(1) Closure*: $\mathcal{S} \subseteq \mathcal{S}^{\perp\perp}$;*
(2) Bi-orthogonal*: $\mathcal{S}^{\perp\perp\perp} = \mathcal{S}^{\perp}$.*

**Generators, Candidates, and Formulas.** The definition of reducibility candidates comes together with a notion of generator, justified by the proposition that follows (which requires an auxiliary lemma).

**Definition 13.4** (Generators and candidates). A *generator* of type $A$ (resp. co-type $A$) is a sub-set $\mathcal{S} \subseteq \mathtt{Terms}_{\vdash A}$ (resp. $\mathcal{S} \subseteq \mathtt{ElCtxs}_A$) such that:

(1) *Non-emptyness*: $\mathcal{S} \neq \emptyset$, and
(2) *Strong normalization*: $\mathcal{S} \subseteq SN_{\to_{\mathrm{ss}}}$.

A generator $\mathcal{S}$ of type $A$ (resp. co-type $A$) is a *candidate* if

(1) *Variables*: $\mathcal{V}_A \subseteq \mathcal{S}$ (resp. $\mathtt{ElVars}_A \subseteq \mathcal{S}$), and
(2) *Bi-orthogonal*: $\mathcal{S} = \mathcal{S}^{\perp\perp}$.

**Lemma 13.5** (Duality and SN). *Let $t \in \mathtt{Terms}_{\vdash A}$ and $E \in \mathtt{ElCtxs}_A$.*

(1) *If $E\langle\!\langle t \rangle\!\rangle \in SN_{\to_{\mathrm{ss}}}$ then $t \in SN_{\to_{\mathrm{ss}}}$ and $E \in SN_{\to_{\mathrm{ss}}}$.*
(2) *If $t \in SN_{\to_{\mathrm{ss}}}$ and $E \in \mathtt{ElVars}_A$ then $E\langle\!\langle t \rangle\!\rangle \in SN_{\to_{\mathrm{ss}}}$.*
(3) *If $E \in SN_{\to_{\mathrm{ss}}}$ and $y \in \mathcal{V}_A$ then $E\langle\!\langle y \rangle\!\rangle \in SN_{\to_{\mathrm{ss}}}$.*

*Proof.*

(1) By induction on $E\langle\!\langle t \rangle\!\rangle \in SN_{\to_{\mathrm{ss}}}$. Let $t = L\langle v \rangle$ and $E = L'\langle [\langle \cdot \rangle \to x] s \rangle$, so that $E\langle\!\langle t \rangle\!\rangle = L'\langle L\langle [v \to x] s \rangle \rangle$. The proof of the statement is based on the obvious fact that every step from $t$ or $L'\langle s \rangle$ (we rely on Remark 13.1) can be mimicked on $L'\langle L\langle [v \to x] s \rangle \rangle$, so that one can then apply the *i.h.*

(2) If $E \in \mathtt{ElVars}_A$ then $E = [\langle \cdot \rangle {\rightarrow} x]x$. Let $t = L\langle v \rangle$. We have to prove that $E \langle\!\langle t \rangle\!\rangle = L\langle [v {\rightarrow} x]x \rangle \in SN_{\rightarrow_{\mathrm{ss}}}$. By induction on $t \in SN_{\rightarrow_{\mathrm{ss}}}$ we show that all the reducts of $s$ are in $SN_{\rightarrow_{\mathrm{ss}}}$. If $E \langle\!\langle t \rangle\!\rangle \rightarrow_{\mathrm{ss}} t$ by reducing the cut on $x$, then the reduct is $t$, which is in $SN_{\rightarrow_{\mathrm{ss}}}$ by hypothesis. Otherwise, $E \langle\!\langle t \rangle\!\rangle$ makes a step in $v$, or in $L$, or involving both $L$ and $v$. The same step can be done on $t$, and thus by *i.h.* the reduct is in $SN_{\rightarrow_{\mathrm{ss}}}$.

(3) Let $L\langle [\langle \cdot \rangle {\rightarrow} x]t \rangle \in E$, so that $E \langle\!\langle y \rangle\!\rangle = L\langle [y {\rightarrow} x]t \rangle$. If $E \langle\!\langle y \rangle\!\rangle \rightarrow_{\mathrm{ss}} L\langle \{y {\rightarrow} x\}t \rangle$ then the reduct is in $SN_{\rightarrow_{\mathrm{ss}}}$ by the hypothesis on $E$. Otherwise, $E \langle\!\langle y \rangle\!\rangle$ makes a step in $t$, or in $L$, or involving both $L$ and $t$. The same step can be done on $L\langle t \rangle$ (we rely on Remark 13.1), and thus by *i.h.* the reduct is in $SN_{\rightarrow_{\mathrm{ss}}}$. $\qquad\square$

**Proposition 13.6.** *If $\mathcal{S}$ is a generator then $\mathcal{S}^\perp$ is a candidate.*

*Proof.* We first consider the case $\mathcal{S} \subseteq \mathtt{Terms}_{\vdash A}$. Properties:

- *Strong normalization*: let $E = L'\langle [\langle \cdot \rangle {\rightarrow} x]s \rangle \in \mathcal{S}^\perp$ and $t = L\langle v \rangle \in \mathcal{S} \neq \emptyset$. By duality, $E \langle\!\langle t \rangle\!\rangle \in SN_{\rightarrow_{\mathrm{ss}}}$. By Lemma 13.5.1, $E \in SN_{\rightarrow_{\mathrm{ss}}}$.
- *Variables* (which subsumes non-emptyness): let $t = L\langle v \rangle \in \mathcal{S}$ and $[\langle \cdot \rangle {\rightarrow} x]x \in \mathtt{ElVars}_A$. By $\mathcal{S} \subseteq SN_{\rightarrow_{\mathrm{ss}}}$ and Lemma 13.5.2 we obtain $L\langle [v {\rightarrow} x]x \rangle \in SN_{\rightarrow_{\mathrm{ss}}}$, that is, $[\langle \cdot \rangle {\rightarrow} x]x \in \mathcal{S}^\perp$.
- *Bi-orthogonal*: by the by-orthogonal property of duality (Lemma 13.3).

Now, the case $\mathcal{S} \subseteq \mathtt{ElCtxs}_A$. Properties:

- *Strong normalization*: let $t = L\langle v \rangle \in \mathcal{S}^\perp$ and $E = L'\langle [\langle \cdot \rangle {\rightarrow} x]s \rangle \in \mathcal{S} \neq \emptyset$. By duality, $E \langle\!\langle t \rangle\!\rangle \in SN_{\rightarrow_{\mathrm{ss}}}$. By Lemma 13.5.1, $t \in SN_{\rightarrow_{\mathrm{ss}}}$.
- *Variables* (which subsumes non-emptyness): let $E \in \mathcal{S}$ and $x \in \mathcal{V}_A$. By $\mathcal{S} \subseteq SN_{\rightarrow_{\mathrm{ss}}}$ and Lemma 13.5.3 we obtain $E \langle\!\langle x \rangle\!\rangle \in SN_{\rightarrow_{\mathrm{ss}}}$, that is, $x \in \mathcal{S}^\perp$.
- *Bi-orthogonal*: by the by-orthogonal property of duality (Lemma 13.3). $\qquad\square$

We now associate to every formula $A$ a set $\mathcal{S}_{[\![A]\!]}$ that we then prove to be a generator, so that its bi-orthogonal is a candidate by Prop. 13.6. A minor unusual point is that we define the set $\mathcal{S}_{[\![X_{\mathtt{m}}]\!]}$ for the atomic formula is the set of multiplicative variables. The literature rather defines it as $\mathtt{Terms}_{X_{\mathtt{m}}} \cap SN_{\rightarrow_{\mathrm{ss}}}$, which is wacky, as this is actually what the method is meant to prove!

**Definition 13.7** (Formulas candidates). Set $[\![A]\!] := \mathcal{S}_{[\![A]\!]}^{\perp\perp}$, where $\mathcal{S}_{[\![A]\!]}$ is defined by induction on $A$:

- $\mathcal{S}_{[\![X_{\mathtt{m}}]\!]} := \mathcal{V}_{\mathtt{m}}$;
- $\mathcal{S}_{[\![A \otimes B]\!]} := \{(t, s) \mid t \in [\![A]\!], s \in [\![B]\!], (t, s) \text{ is proper}\}$;
- $\mathcal{S}_{[\![A \multimap B]\!]} := \left\{ \lambda x.t \;\middle|\; \begin{array}{l} \Gamma, x : A \vdash t : B, \\ L\langle [v {\rightarrow} x]t \rangle \in [\![B]\!] \; \forall s = L\langle v \rangle \in [\![A]\!], \text{ and} \\ \lambda x.t \text{ is proper} \end{array} \right\}$;
- $\mathcal{S}_{[\![!A]\!]} := \{!t \mid t \in [\![A]\!], !t \text{ is proper}\}$.

The proof of that $[\![A]\!]$ is a candidate for every formula $A$ requires a lemma for the lolli case.

**Lemma 13.8.** *If $[\![A]\!] \subseteq SN_{\rightarrow_{\mathrm{ss}}}$ and $\mathcal{V}_B \subset [\![B]\!]$ then $\mathcal{S}_{[\![A \multimap B]\!]} \neq \emptyset$.*

*Proof.* We show that $\lambda x.(\!|m {\oslash} x, y|\!)y \in \mathcal{S}_{[\![A \multimap B]\!]}$. The idea is that the body $(\!|m {\oslash} x, y|\!)y$ of that abstraction is the smallest term $t$ verifying the typing requirement $\Gamma, x : A \vdash t : B$ when $A \neq B$. By definition, $\lambda x.(\!|m {\oslash} x, y|\!)y \in \mathcal{S}_{[\![A \multimap B]\!]}$ holds if $L\langle [v {\rightarrow} x](\!|m {\oslash} x, y|\!)y \rangle \in [\![B]\!]$ for every $L\langle v \rangle \in [\![A]\!]$, that is, $E \langle\!\langle L\langle [v {\rightarrow} x](\!|m {\oslash} x, y|\!)y \rangle \rangle\!\rangle \in SN_{\rightarrow_{\mathrm{ss}}}$ for every $E \in [\![B]\!]^\perp$. By hypothesis,

$\mathcal{V}_B \subseteq \llbracket B \rrbracket$, and duality gives $E\langle\!\langle y \rangle\!\rangle \in SN_{\to_{ss}}$. By hypothesis, we also have $L\langle v \rangle \in SN_{\to_{ss}}$. The proof is in two steps:

- Proving $t := E\langle\!\langle L\langle (\!|m\otimes v, y|\!) y \rangle \rangle\!\rangle \in SN_{\to_{ss}}$: note that the two components $L\langle v \rangle$ and $E\langle\!\langle y \rangle\!\rangle$ cannot interact in any way, so that every step of $t$ comes from a step of either $L\langle v \rangle$ or $E\langle\!\langle y \rangle\!\rangle$, as in the proof of the extension property. Formally, the proof is by induction on $(L\langle v \rangle \in SN_{\to_{ss}}, E\langle\!\langle y \rangle\!\rangle \in SN_{\to_{ss}})$.
- Proving $s := E\langle\!\langle L\langle [v\to x](\!|m\otimes x, y|\!) y \rangle \rangle\!\rangle \in SN_{\to_{ss}}$: note that $t = E\langle\!\langle L\langle (\!|m\otimes v, y|\!) y \rangle \rangle\!\rangle = \{v\to x\}E\langle\!\langle L\langle (\!|m\otimes x, y|\!) y \rangle \rangle\!\rangle$, thus by root cut expansion we obtain $[v\to x]E\langle\!\langle L\langle (\!|m\otimes x, y|\!) y \rangle \rangle\!\rangle \in SN_{\to_{ss}}$. By structural stability, $[v\to x]E\langle\!\langle L\langle (\!|m\otimes x, y|\!) y \rangle \rangle\!\rangle \equiv_{\mathtt{cut}} s \in SN_{\to_{ss}}$.

Therefore, $\lambda x.(\!|m\otimes x, y|\!) y \in \mathcal{S}_{\llbracket A \multimap B \rrbracket} \neq \emptyset$. $\qquad\square$

The proof of the next proposition is simple and yet tricky: by induction on $A$, it uses Prop. 13.6 to prove 2 from 1, and 3 from 2, but it also needs 3 (on sub-formulas) to prove 1.

**Proposition 13.9** (Formulas induce reducibility candidates)**.** *Let $A$ be a IMELL formula.*

(1) *Generators:* $\mathcal{S}_{\llbracket A \rrbracket}$ *is a generator.*
(2) *Dual candidates:* $\llbracket A \rrbracket^\perp$ *is a candidate.*
(3) *Candidates:* $\llbracket A \rrbracket$ *is a candidate.*

*Proof.* We prove the first point, the second follows from the first and Prop. 13.6, the third one follows from the second and Prop. 13.6. By induction on $A$. Each case of the proof shall use the extension property (Lemma 12.1). Cases:

- *Base*, i.e. $A = X_{\mathfrak{m}}$. Note that $\mathcal{S}_{\llbracket X_{\mathfrak{m}} \rrbracket}$ is non-empty by definition and that all its elements are normal.
- *Tensor*, i.e. $A = B \otimes C$. By point 3 of the *i.h.*, $\llbracket B \rrbracket, \llbracket C \rrbracket \subseteq SN_{\to_{ss}}$ and they are non-empty. Then $\mathcal{S}_{\llbracket B \otimes C \rrbracket}$ is non-empty. By extension, $\mathcal{S}_{\llbracket B \otimes C \rrbracket} \subseteq SN_{\to_{ss}}$.
- *Implication*, i.e. $A = B \multimap C$. Proving that $\mathcal{S}_{\llbracket B \multimap C \rrbracket}$ is non-empty requires the previous lemma, because by definition it does not contains the variables of type $B \multimap C$, nor it is defined by abstracting terms in $\llbracket C \rrbracket$. By *i.h.* (point 3) applied to $C$, $\mathcal{V}_C \subseteq \llbracket C \rrbracket$, and By *i.h.* (point 3) applied to $B$, $L\langle v \rangle \in SN_{\to_{ss}}$. Then by Lemma 13.8, $\mathcal{S}_{\llbracket B \multimap C \rrbracket} \neq \emptyset$.
  Since variables are in $\llbracket B \rrbracket$ (by Point 3 of the *i.h.*), if $\lambda x.u \in \mathcal{S}_{\llbracket B \multimap C \rrbracket}$ then $[x\to x]u \in \llbracket C \rrbracket$. By *i.h.* (Point 3), $[x\to x]u \in SN_{\to_{ss}}$ and so does $u$. By extension, $\lambda x.u \in SN_{\to_{ss}}$. Therefore $\mathcal{S}_{\llbracket B \multimap C \rrbracket} \subseteq SN_{\to_{ss}}$.
- *Bang*, i.e. $A = !B$. By point 3 of the *i.h.*, $\llbracket B \rrbracket \subseteq SN_{\to_{ss}}$ and it is non-empty. Then $\mathcal{S}_{\llbracket !B \rrbracket}$ is non-empty. By extension, $\mathcal{S}_{\llbracket !B \rrbracket} \subseteq SN_{\to_{ss}}$. $\qquad\square$

**Reducibility and Adequacy.** In the following notion of reducible derivations, the rigid structure of terms forces an order between the assignments in the typing context $\Gamma$, that is, it treats $\Gamma$ as a *list* rather than as a multi-set. The proof of adequacy then considers that the sequent calculus comes with an exchange rule, treated by one of the cases.

**Definition 13.10** (Reducible derivations)**.** Let $\pi \rhd \Gamma \vdash t : A$ be a typing derivation, and $\Gamma = x_1 : B_1, .., x_k : B_k$. Then $\pi$ is *reducible* if $[v_1\to x_1] \ldots [v_k\to x_k]t \in \llbracket A \rrbracket$ for every value $v_i \in \mathcal{S}_{\llbracket B_i \rrbracket}$ such that the introduced cuts are independent, that is, $\mathtt{fv}(v_i) \cap \mathtt{dom}(\Gamma) = \emptyset$. To ease notations, we shorten $[v_1\to x_1] \ldots [v_k\to x_k]t$ to $[v_i\to x_i]_\Gamma t$.

Before proving adequacy, we provide an equivalent but slightly extended reformulation of the reducibility clause, in which the variables $x_i$ are cut with arbitrary terms, not necessarily with values. The variant is used in the below proof of adequacy when dealing with: the left rules having $L\langle v\rangle$ as hypothesis (cut and subtraction) or the rewriting rule of which has to split terms on-the-fly (par and dereliction), and in the lolli case, because split terms are used in the definition of $\mathcal{S}_{[\![A\multimap B]\!]}$. *Notations*: if $s = L\langle v\rangle$, we use $[\langle s\rangle\to x]t$ for $L\langle[v\to x]t\rangle$, and $[\langle s_i\rangle\to x_i]_\Gamma t$ for $[\langle s_1\rangle\to x_1]\ldots[\langle s_k\rangle\to x_k]t$.

**Lemma 13.11** (Extended reducibility clause). *Let $\pi \vartriangleright \Gamma \vdash t : A$ with $\Gamma = x_1 : B_1, .., x_k : B_k$ be a typed derivation. Then $\pi$ is* reducible *if and only if $[\langle s\rangle\to x_i]_\Gamma t \in [\![A]\!]$ for every $s \in [\![B_i]\!]$ such that the introduced cuts are independent.*

*Proof.* Direction $\Leftarrow$ is obvious because $\mathcal{S}_{[\![B_i]\!]} \subseteq [\![B_i]\!]$, we prove direction $\Rightarrow$. Let $\Gamma = \Delta, x : B, \Pi$. The hypothesis is

$$E\langle\!\langle[v_i\to y_i]_\Pi[v\to x][v_j\to z_j]_\Delta t\rangle\!\rangle \in SN_{\to_{ss}} \tag{13.1}$$

for every $v \in \mathcal{S}_{[\![B]\!]}$ and appropriate $v_i$ and $v_j$. We show that we can replace $v$ with $s \in [\![B]\!]$, that is, that the following holds:

$$E\langle\!\langle[v_i\to y_i]_\Pi[\langle s\rangle\to x][v_j\to z_j]_\Delta t\rangle\!\rangle \in SN_{\to_{ss}} \tag{13.2}$$

By iterating the reasoning on all other values $v_i$ and $v_j$ one obtains the statement. Since (13.1) holds for all $v \in \mathcal{S}_{[\![B]\!]}$ we have that $E' := E\langle\!\langle[v_i\to y_i]_\Pi[\langle\cdot\rangle\to x][v_j\to z_j]_\Delta t\rangle\!\rangle \in \mathcal{S}_{[\![B]\!]}^\perp = [\![B]\!]^\perp$. Note that the cuts $[v_i\to y_i]_\Pi$ and $[v_j\to z_j]_\Delta$ can be added to the elimination context exactly because they are independent. Now, by duality we obtain $E'\langle\!\langle s\rangle\!\rangle \in SN_{\to_{ss}}$ for every $s \in [\![B]\!]$, which is exactly (13.2). □

**Theorem 13.12** (Adequacy). *Let $\pi \vartriangleright \Gamma \vdash t : A$ a type derivation. Then $\pi$ is reducible.*

*Proof.* By induction on $\pi$. The proof rests on the three rewriting properties mentioned at the beginning of the section, namely extension, root cut expansion, and structural stability. Let $\Gamma = x_1 : B_1, .., x_k : B_k$. We show the most relevant cases, the other ones are in Appendix A, page 63. Cases of the last rule of $\pi$:

- *Exponential axiom*:

$$\frac{}{e : !A \vdash e : !A}\ \mathsf{ax_e}$$

  We need to show that $E\langle\!\langle[!s\to e]e\rangle\!\rangle \in SN_{\to_{ss}}$ for every term $s \in [\![A]\!] \subseteq SN_{\to_{ss}}$ and every elimination context $E \in [\![!A]\!]^\perp$. Note that $E\langle\!\langle[!s\to e]e\rangle\!\rangle \to_! E\langle\!\langle !s\rangle\!\rangle$ which is in $SN_{\to_{ss}}$ by duality. Note also that $E\langle\!\langle !s\rangle\!\rangle = E\langle\!\langle\{!s\to e\}e\rangle\!\rangle = \{!s\to e\}E\langle\!\langle e\rangle\!\rangle$. By extension, $!s \in SN_{\to_{ss}}$. Since both $!s$ and $\{!s\to e\}E\langle\!\langle e\rangle\!\rangle$ are in $SN_{\to_{ss}}$, root cut expansion gives $[!s\to e]E\langle\!\langle e\rangle\!\rangle \in SN_{\to_{ss}}$. By structural stability, $[!s\to e]E\langle\!\langle e\rangle\!\rangle \equiv_{\mathsf{cut}} E\langle\!\langle[!s\to e]e\rangle\!\rangle \in SN_{\to_{ss}}$.

  Note that the two terms $[!s\to e]E\langle\!\langle e\rangle\!\rangle$ and $E\langle\!\langle[!s\to e]e\rangle\!\rangle$ do translate to the same proof net, which is why structural stability does not appear in the proofs of SN for proof nets. In the intuitionistic case with weakenings, however, one needs some form of jumps and jump rewiring rules, so an analogous of structural stability would be most probably needed. There are no proofs of SN for intuitionistic proof nets in the literature.

- *Exponential inductive cases.* We show the promotion and contraction cases, the weakening and dereliction cases are minor variations over the contraction one.
  - *Promotion*:

$$\frac{\rho \vartriangleright !\Delta \vdash s : C}{!\Delta \vdash !s : !C}\ !_r$$

With $t = {!}s$ and $\Gamma = {!}\Delta$. Let ${!}\Delta = e_1 : {!}B_1, \ldots, e_k : {!}B_k$. We need to prove that $[{!}u_i{\to}e_i]_{!\Delta}{!}s \in \llbracket{!}C\rrbracket$ for every term $u_i \in \llbracket B_i\rrbracket$, that is, $E\langle\!\langle[{!}u_i{\to}e_i]_{!\Delta}{!}s\rangle\!\rangle \in SN_{\to_{ss}}$ for every elimination context $E \in \llbracket{!}C\rrbracket^\perp$. By $i.h.$, $\rho$ is reducible, therefore $[{!}u_i{\to}e_i]_{!\Delta}s \in \llbracket C\rrbracket$, and so ${!}[{!}u_i{\to}e_i]_{!\Delta}s \in \llbracket{!}C\rrbracket$, that is, $E\langle\!\langle{!}[{!}u_i{\to}e_i]_{!\Delta}s\rangle\!\rangle \in SN_{\to_{ss}}$. Note that we have $E\langle\!\langle{!}[{!}u_i{\to}e_i]_{!\Delta}s\rangle\!\rangle \to^*_{e^{ss}} E\langle\!\langle{!}\{{!}u_i{\to}e_i\}_{!\Delta}s\rangle\!\rangle$ which is equal to $E\langle\!\langle\{{!}u_i{\to}e_i\}_{!\Delta}{!}s\rangle\!\rangle = \{{!}u_i{\to}e_i\}_{!\Delta}E\langle\!\langle{!}s\rangle\!\rangle$ which is then in $SN_{\to_{ss}}$. By hypothesis, $u_i \in \llbracket B_i\rrbracket$, which implies $u_i \in SN_{\to_{ss}}$ by the properties of candidates (Prop. 13.9), thus ${!}u_i \in SN_{\to_{ss}}$ by extension. By root cut expansion, $[{!}u_i{\to}e_i]_{!\Delta}E\langle\!\langle{!}s\rangle\!\rangle \in SN_{\to_{ss}}$. By structural stability, $[{!}u_i{\to}e_i]_{!\Delta}E\langle\!\langle{!}s\rangle\!\rangle \equiv_{\mathsf{cut}} E\langle\!\langle[{!}u_i{\to}e_i]_{!\Delta}{!}s\rangle\!\rangle \in SN_{\to_{ss}}$.

– *Contraction*:
$$\frac{\Delta, e : {!}B, f : {!}B \vdash s : A}{\Delta, e : {!}B \vdash \{e{\to}f\}s : A}\ \mathsf{c}$$
with $t = \{e{\to}f\}s$ and $\Gamma = \Delta, e : {!}B$. We use the notation $x_i : B_i$ for the assignments in $\Delta$. By $i.h.$, $s' := E\langle\!\langle[v_i{\to}x_i]_\Delta[{!}u{\to}e][{!}u{\to}f]s\rangle\!\rangle \in SN_{\to_{ss}}$ for every $u \in \llbracket B\rrbracket$, every $v_i \in \mathcal{S}_{\llbracket B_i\rrbracket}$ for $i \in \{1, \ldots, k\}$, and every $E \in \llbracket A\rrbracket^\perp$. We have:
$$s' \to^*_{e^{ss}} E\langle\!\langle[v_i{\to}x_i]_\Delta\{{!}u{\to}e\}\{{!}u{\to}f\}s\rangle\!\rangle \in SN_{\to_{ss}}.$$
By the properties of meta-level substitution (namely Lemma 6.5.8), we have
$$E\langle\!\langle[v_i{\to}x_i]_\Delta\{{!}u{\to}e\}\{{!}u{\to}f\}s\rangle\!\rangle = E\langle\!\langle[v_i{\to}x_i]_\Delta\{{!}u{\to}e\}\{e{\to}f\}s\rangle\!\rangle$$
which is equal to $E\langle\!\langle\{{!}u{\to}e\}[v_i{\to}x_i]_\Delta\{e{\to}f\}s\rangle\!\rangle$ by the independence of cuts in the definition of reducibility, in turn equal to $\{{!}u{\to}e\}E\langle\!\langle[v_i{\to}x_i]_\Delta\{e{\to}f\}s\rangle\!\rangle$. Since $u \in \llbracket B\rrbracket$, we have $u \in SN_{\to_{ss}}$ by the properties of candidates (Prop. 13.9), thus ${!}u \in SN_{\to_{ss}}$ by extension. By root cut expansion, $[{!}u{\to}e]E\langle\!\langle[v_i{\to}x_i]_\Delta\{e{\to}f\}s\rangle\!\rangle \in SN_{\to_{ss}}$. By structural stability,
$$[{!}u{\to}e]E\langle\!\langle[v_i{\to}x_i]_\Delta\{e{\to}f\}s\rangle\!\rangle \equiv_{\mathsf{cut}} E\langle\!\langle[v_i{\to}x_i]_\Delta\{e{\to}f\}[{!}u{\to}e]s\rangle\!\rangle \in SN_{\to_{ss}}$$
which is exactly what is required.

- *Multiplicative inductive cases*. We show the lolli and subtraction cases. The subtraction case, in particular, combines the reasoning at work in both the simpler cut and par cases, which are in the appendix.
– *Lolli*:
$$\frac{\rho \rhd x : B, \Gamma \vdash s : C}{\Gamma \vdash \lambda x.s : B \multimap C}\ \multimap_r$$
with $t = \lambda x.s$ and $A = B \multimap C$. We have to show that $t' := E\langle\!\langle[v_i{\to}x_i]_\Gamma\lambda x.s\rangle\!\rangle \in SN_{\to_{ss}}$ for every $v_i \in \mathcal{S}_{\llbracket B_i\rrbracket}$ for $i \in \{1, \ldots, k\}$ and every $E \in \llbracket B \multimap C\rrbracket^\perp$. By $i.h.$ on $\rho$ and the extended reducibility clause (Lemma 13.11), we have $L'\langle[v{\to}x][v_i{\to}x_i]_\Gamma s\rangle \in \llbracket C\rrbracket$ for every $L'\langle v\rangle \in \llbracket B\rrbracket$ and with $x \notin \mathtt{fv}(v_i)$ for all $i$ because of the independence of cuts in the definition of reducibility. Then $\lambda x.[v_i{\to}x_i]_\Gamma s \in \llbracket B \multimap C\rrbracket$. By duality, $E\langle\!\langle\lambda x.[v_i{\to}x_i]_\Gamma s\rangle\!\rangle \in SN_{\to_{ss}}$. By structural stability and $x \notin \mathtt{fv}(v_i)$,
$$E\langle\!\langle\lambda x.[v_i{\to}x_i]_\Gamma s\rangle\!\rangle \equiv_{\mathsf{cut}} E\langle\!\langle[v_i{\to}x_i]_\Gamma\lambda x.s\rangle\!\rangle = t' \in SN_{\to_{ss}}.$$
– *Subtraction*:
$$\frac{\pi_l \rhd \Gamma_l \vdash L\langle v\rangle : C \qquad \pi_r \rhd \Gamma_r, x : B \vdash s : A \qquad \Gamma\#(\Gamma_r, x : B),\ m\ \text{fresh}}{\Gamma_l, \Gamma_r, m : C \multimap B \vdash L\langle(\!|m\otimes v, x|\!)s\rangle : A}\ \multimap_l$$
with $t = L\langle(\!|m\otimes v, x|\!)s\rangle$ and $\Gamma = \Gamma_l, \Gamma_r, m : C \multimap B$. This case combines the reasoning for the unary left rules with the one for cut. We have to show that
$$t' := E\langle\!\langle[v_i{\to}x_i]_\Gamma[\lambda y.r{\to}m]L\langle(\!|m\otimes v, x|\!)s\rangle\rangle\!\rangle \in SN_{\to_{ss}}$$
for every $v_i \in \mathcal{S}_{\llbracket B_i\rrbracket}$ for $i \in \{1, \ldots, k\}$, every $\lambda y.r \in \mathcal{S}_{\llbracket C\multimap B\rrbracket}$, and every $E \in \llbracket A\rrbracket^\perp$. By $i.h.$ on $\pi_l$, we have $[v_i{\to}x_i]_{\Gamma_l}L\langle v\rangle \in \llbracket C\rrbracket$. By definition of $\lambda y.r \in \mathcal{S}_{\llbracket C\multimap B\rrbracket}$, we obtain $[v_i{\to}x_i]_{\Gamma_l}L\langle[v{\to}y]r\rangle \in \llbracket B\rrbracket$. Let $r = L'\langle v'\rangle$. Then $[v_i{\to}x_i]_{\Gamma_l}L\langle[v{\to}y]L'\langle v'\rangle\rangle \in$

$[\![B]\!]$. By *i.h.* $\pi_r$ is reducible. By the extended reducibility clause (Lemma 13.11), $[v_i{\to}x_i]_{\Gamma_r}[v_i{\to}x_i]_{\Gamma_l}L\langle[v{\to}y]L'\langle[v'{\to}x]s\rangle\rangle \in [\![A]\!]$, that is,

$$
\begin{aligned}
E\langle\!\langle[v_i{\to}x_i]_{\Gamma_r}[v_i{\to}x_i]_{\Gamma_l}L\langle[v{\to}y]L'\langle[v'{\to}x]s\rangle\rangle\rangle\!\rangle &= \\
E\langle\!\langle[v_i{\to}x_i]_{\Gamma}L\langle[v{\to}y]L'\langle[v'{\to}x]s\rangle\rangle\rangle\!\rangle &\in SN_{\to_{\mathrm{ss}}}.
\end{aligned}
$$

By root cut expansion, $[\lambda y.L'\langle v'\rangle{\to}m]E\langle\!\langle[v_i{\to}x_i]_{\Gamma}L\langle(\!|m\oslash v,x|\!)s\rangle\rangle\!\rangle \in SN_{\to_{\mathrm{ss}}}$. By structural stability, $[\lambda y.L'\langle v'\rangle{\to}m]E\langle\!\langle[v_i{\to}x_i]_{\Gamma}L\langle(\!|m\oslash v,x|\!)s\rangle\rangle\!\rangle \equiv_{\mathsf{cut}} t' \in SN_{\to_{\mathrm{ss}}}$.  □

**Corollary 13.13** (Typable terms are SN). *Let $t$ be a typable term. Then $t \in SN_{\to_{\mathrm{ss}}}$ and $t \in SN_{\to_{\mathrm{ms}}}$.*

*Proof.* Since $t$ is typable, we have $\pi \vartriangleright x_1 : B_1, \ldots, x_k : B_k \vdash t : A$, for some derivation $\pi$. By adequacy (Theorem 13.12), $\pi$ is reducible. By Prop. 13.9, $x_i \in [\![B_i]\!]$ for $i \in \{1, \ldots, k\}$ and $[\langle\cdot\rangle{\to}y]y \in [\![A]\!]^{\perp}$. Let $t = L\langle v\rangle$. By reducibility of $\pi$, $s := [x_1{\to}x_1]\ldots[x_k{\to}x_k]L\langle[v{\to}y]y\rangle \in SN_{\to_{\mathrm{ss}}}$. Note that $s \to^*_{\mathsf{ss}} t$, thus $t \in SN_{\to_{\mathrm{ss}}}$. By PSN (Theorem 12.3), $t \in SN_{\to_{\mathrm{ms}}}$.  □

## 14. CONCLUSIONS

This work lifts Accattoli and Kesner's *linear substitution calculus* to the sequent calculus for IMELL via 3 generalizations: adding *explicit (non-)linearity*, a simple form of *pattern matching* ($\otimes$), and replacing applications with *subtractions*.

We provide 3 main contributions for the new *exponential substitution calculus*. Firstly, we show that IMELL untyped exponentials, in contrast to the classical case, are strongly normalizing. Secondly, we define the *good evaluation strategy* and show that it has the subterm property, obtaining the *first* polynomial cost model for IMELL. Thirdly, we provide elegant proofs of confluence, strong normalization, and preservation of strong normalization.

Methodologically, we provide an extensive study of linear logic cut elimination that is *not* based on proof nets.

**The Next Step.** In the $\lambda$-calculus, the number of (leftmost) $\beta$-steps is a polynomial cost model [ADL16]. Since *one* $\beta$-steps is simulated in IMELL by *one* multilicative step (actually a $\to_{\multimap}$ step) followed by possibly *many* exponential steps, it means that one can count only the number of multiplicative (or even $\to_{\multimap}$) steps, that is, *one can count 0 for exponential steps*. Such a surprising fact is enabled by a sophisticated technique called *useful sharing* [ADL16, ACSC21, AL22]. Can useful sharing be generalized to the ESC as to turn the number of multiplicative/$\to_{\multimap}$ good steps into a polynomial time cost model? For this, termination of untyped exponentials is mandatory, thus, the results of this paper suggest that it might be possible. Still, the question is far from obvious, as in the $\lambda$-calculus there is a strong, hardcoded correlation between multiplicatives and exponentials, not present in IMELL. In the standard call-by-name/value encodings of $\lambda$-calculus in IMELL, indeed, multiplicatives and exponentials connectives rigidly alternate, while IMELL enables consecutive exponentials, as in $!!A$, allowing wilder exponential behavior.

**Further Future Work.** Beyond refining the cost model, there are many possible directions for future work. Is the good strategy normalizing on *untyped* (clash-free) terms (that is, does it reach a cut-free reduct whenever there is one)? Good redexes are not stable by cut/left-equivalence: is there a generalization that is stable? Is it possible to measure good steps via relational semantics/multi types, refining the work of de Carvalho et al. [dCPTdF11]? Being a generalization of linear head reduction, does the good strategy have

a similar connection to game semantics (see [DHR96, Cla15])? Can the good strategy be exploited for implicit computational complexity? Can the *left context/value* splitting be given a logical status? What about a LSC/cost-aware approach for the additives?

## Acknowledgments

## References

[ABKL14]   Beniamino Accattoli, Eduardo Bonelli, Delia Kesner, and Carlos Lombardi. A nonstandard standardization theorem. In Suresh Jagannathan and Peter Sewell, editors, *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 659–670. ACM, 2014. `doi:10.1145/2535838.2535886`.

[ABM14]   Beniamino Accattoli, Pablo Barenbaum, and Damiano Mazza. Distilling abstract machines. In Johan Jeuring and Manuel M. T. Chakravarty, editors, *Proceedings of the 19th ACM SIGPLAN international conference on Functional programming, Gothenburg, Sweden, September 1-3, 2014*, pages 363–376. ACM, 2014. `doi:10.1145/2628136.2628154`.

[ABM15]   Beniamino Accattoli, Pablo Barenbaum, and Damiano Mazza. A strong distillery. In Xinyu Feng and Sungwoo Park, editors, *Programming Languages and Systems - 13th Asian Symposium, APLAS 2015, Pohang, South Korea, November 30 - December 2, 2015, Proceedings*, volume 9458 of *Lecture Notes in Computer Science*, pages 231–250. Springer, 2015. `doi:10.1007/978-3-319-26529-2\_13`.

[Abr93]   Samson Abramsky. Computational interpretations of linear logic. *Theor. Comput. Sci.*, 111(1&2):3–57, 1993. `doi:10.1016/0304-3975(93)90181-R`.

[Acc12]   Beniamino Accattoli. An abstract factorization theorem for explicit substitutions. In Ashish Tiwari, editor, *23rd International Conference on Rewriting Techniques and Applications (RTA'12) , RTA 2012, May 28 - June 2, 2012, Nagoya, Japan*, volume 15 of *LIPIcs*, pages 6–21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012. `doi:10.4230/LIPIcs.RTA.2012.6`.

[Acc13]   Beniamino Accattoli. Linear logic and strong normalization. In Femke van Raamsdonk, editor, *24th International Conference on Rewriting Techniques and Applications, RTA 2013, June 24-26, 2013, Eindhoven, The Netherlands*, volume 21 of *LIPIcs*, pages 39–54. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013. `doi:10.4230/LIPIcs.RTA.2013.39`.

[Acc18]   Beniamino Accattoli. Proof nets and the linear substitution calculus. In Bernd Fischer and Tarmo Uustalu, editors, *Theoretical Aspects of Computing - ICTAC 2018 - 15th International Colloquium, Stellenbosch, South Africa, October 16-19, 2018, Proceedings*, volume 11187 of *Lecture Notes in Computer Science*, pages 37–61. Springer, 2018. `doi:10.1007/978-3-030-02508-3\_3`.

[Acc22]   Beniamino Accattoli. Exponentials as substitutions and the cost of cut elimination in linear logic (v1), 2022. URL: `https://arxiv.org/abs/2205.15203v1`.

[ACCL91]   Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. *J. Funct. Program.*, 1(4):375–416, 1991. `doi:10.1017/S0956796800000186`.

[ACGSC19]   Beniamino Accattoli, Andrea Condoluci, Giulio Guerrieri, and Claudio Sacerdoti Coen. Crumbling abstract machines. In Ekaterina Komendantskaya, editor, *Proceedings of the 21st International Symposium on Principles and Practice of Programming Languages, PPDP 2019, Porto, Portugal, October 7-9, 2019*, pages 4:1–4:15. ACM, 2019. `doi:10.1145/3354166.3354169`.

[ACM00]   Andrea Asperti, Paolo Coppola, and Simone Martini. (optimal) duplication is not elementary recursive. In *POPL 2000*, pages 96–107, 2000. URL: `http://doi.acm.org/10.1145/325694.325707`, `doi:10.1145/325694.325707`.

[ACSC21]    Beniamino Accattoli, Andrea Condoluci, and Claudio Sacerdoti Coen. Strong call-by-value is reasonable, implosively. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–14. IEEE, 2021. `doi:10.1109/LICS52264.2021.9470630`.

[AD17]    Pablo Arrighi and Gilles Dowek. Lineal: A linear-algebraic lambda-calculus. *Log. Methods Comput. Sci.*, 13(1), 2017. `doi:10.23638/LMCS-13(1:8)2017`.

[ADL12]    Beniamino Accattoli and Ugo Dal Lago. On the invariance of the unitary cost model for head reduction. In Ashish Tiwari, editor, *23rd International Conference on Rewriting Techniques and Applications (RTA'12) , RTA 2012, May 28 - June 2, 2012, Nagoya, Japan*, volume 15 of *LIPIcs*, pages 22–37. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012. `doi:10.4230/LIPIcs.RTA.2012.22`.

[ADL16]    Beniamino Accattoli and Ugo Dal Lago. (Leftmost-Outermost) Beta-Reduction is Invariant, Indeed. *Logical Methods in Computer Science*, 12(1), 2016. `doi:10.2168/LMCS-12(1:4)2016`.

[AFM+95]    Zena M. Ariola, Matthias Felleisen, John Maraist, Martin Odersky, and Philip Wadler. The call-by-need lambda calculus. In *22nd ACM Symposium on Principles of Programming Languages, POPL'95*, pages 233–246. ACM Press, 1995. `doi:10.1145/199448.199507`.

[AK10]    Beniamino Accattoli and Delia Kesner. The structural *lambda*-calculus. In Anuj Dawar and Helmut Veith, editors, *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 381–395. Springer, 2010. `doi:10.1007/978-3-642-15205-4\_30`.

[AL22]    Beniamino Accattoli and Maico Leberle. Useful open call-by-need. In Florin Manea and Alex Simpson, editors, *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*, volume 216 of *LIPIcs*, pages 4:1–4:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPIcs.CSL.2022.4`.

[AP12]    Beniamino Accattoli and Luca Paolini. Call-by-value solvability, revisited. In Tom Schrijvers and Peter Thiemann, editors, *Functional and Logic Programming - 11th International Symposium, FLOPS 2012, Kobe, Japan, May 23-25, 2012. Proceedings*, volume 7294 of *Lecture Notes in Computer Science*, pages 4–16. Springer, 2012. `doi:10.1007/978-3-642-29822-6\_4`.

[Asp98]    Andrea Asperti. Light affine logic. In *Thirteenth Annual IEEE Symposium on Logic in Computer Science, Indianapolis, Indiana, USA, June 21-24, 1998*, pages 300–308. IEEE Computer Society, 1998. `doi:10.1109/LICS.1998.705666`.

[BB17]    Pablo Barenbaum and Eduardo Bonelli. Optimality and the linear substitution calculus. In Dale Miller, editor, *2nd International Conference on Formal Structures for Computation and Deduction, FSCD 2017, September 3-9, 2017, Oxford, UK*, volume 84 of *LIPIcs*, pages 9:1–9:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. `doi:10.4230/LIPIcs.FSCD.2017.9`.

[BBdPH93]    P. N. Benton, Gavin M. Bierman, Valeria de Paiva, and Martin Hyland. A term calculus for intuitionistic linear logic. In Marc Bezem and Jan Friso Groote, editors, *Typed Lambda Calculi and Applications, International Conference on Typed Lambda Calculi and Applications, TLCA '93, Utrecht, The Netherlands, March 16-18, 1993, Proceedings*, volume 664 of *Lecture Notes in Computer Science*, pages 75–90. Springer, 1993. `doi:10.1007/BFb0037099`.

[BCD21]    Malgorzata Biernacka, Witold Charatonik, and Tomasz Drab. A derived reasonable abstract machine for strong call by value. In Niccolò Veltri, Nick Benton, and Silvia Ghilezan, editors, *PPDP 2021: 23rd International Symposium on Principles and Practice of Declarative Programming, Tallinn, Estonia, September 6-8, 2021*, pages 6:1–6:14. ACM, 2021. `doi:10.1145/3479394.3479401`.

[Ben94]    P. N. Benton. A mixed linear and non-linear logic: Proofs, terms and models (extended abstract). In Leszek Pacholski and Jerzy Tiuryn, editors, *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers*, volume 933 of *Lecture Notes in Computer Science*, pages 121–135. Springer, 1994. `doi:10.1007/BFb0022251`.

[Ben95]    P. N. Benton. Strong normalisation for the linear term calculus. *J. Funct. Program.*, 5(1):65–80, 1995. `doi:10.1017/S0956796800001246`.

[BG95] Guy E. Blelloch and John Greiner. Parallelism in sequential functional languages. In John Williams, editor, *Proceedings of the seventh international conference on Functional programming languages and computer architecture, FPCA 1995, La Jolla, California, USA, June 25-28, 1995*, pages 226–237. ACM, 1995. `doi:10.1145/224164.224210`.

[BKR21] Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca. Solvability = typability + inhabitation. *Log. Methods Comput. Sci.*, 17(1), 2021. URL: `https://lmcs.episciences.org/7141`.

[BS94] Gianluigi Bellin and Philip J. Scott. On the pi-calculus and linear logic. *Theor. Comput. Sci.*, 135(1):11–65, 1994. `doi:10.1016/0304-3975(94)00104-9`.

[BW96] P. N. Benton and Philip Wadler. Linear logic, monads and the lambda calculus. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 420–431. IEEE Computer Society, 1996. `doi:10.1109/LICS.1996.561458`.

[CA21] Jules Chouquet and Lionel Vaux Auclair. An application of parallel cut elimination in multiplicative linear logic to the taylor expansion of proof nets. *Log. Methods Comput. Sci.*, 17(4), 2021. `doi:10.46298/lmcs-17(4:22)2021`.

[CFM16] Pierre-Louis Curien, Marcelo P. Fiore, and Guillaume Munch-Maccagnoni. A theory of effects and resources: adjunction models and polarised calculi. In Rastislav Bodík and Rupak Majumdar, editors, *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pages 44–56. ACM, 2016. `doi:10.1145/2837614.2837652`.

[CG14] Alberto Carraro and Giulio Guerrieri. A semantical and operational account of call-by-value solvability. In Anca Muscholl, editor, *Foundations of Software Science and Computation Structures - 17th International Conference, FOSSACS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, volume 8412 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 2014. `doi:10.1007/978-3-642-54830-7\_7`.

[CH00] Pierre-Louis Curien and Hugo Herbelin. The duality of computation. In Martin Odersky and Philip Wadler, editors, *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00), Montreal, Canada, September 18-21, 2000*, pages 233–243. ACM, 2000. `doi:10.1145/351240.351262`.

[CK97] Roberto Di Cosmo and Delia Kesner. Strong normalization of explicit substitutions via cut elimination in proof nets (extended abstract). In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 - July 2, 1997*, pages 35–46. IEEE Computer Society, 1997. `doi:10.1109/LICS.1997.614927`.

[Cla15] Pierre Clairambault. Bounding linear head reduction and visible interaction through skeletons. *Log. Methods Comput. Sci.*, 11(2), 2015. `doi:10.2168/LMCS-11(2:6)2015`.

[CP10] Luís Caires and Frank Pfenning. Session types as intuitionistic linear propositions. In Paul Gastin and François Laroussinie, editors, *CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings*, volume 6269 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 2010. `doi:10.1007/978-3-642-15375-4\_16`.

[Dan90] Vincent Danos. *La Logique Linéaire appliqué à l'étude de divers processus de normalisation (principalement du λ-calcul)*. Phd thesis, Université Paris 7, 1990.

[dCPTdF11] Daniel de Carvalho, Michele Pagani, and Lorenzo Tortora de Falco. A semantic measure of the execution time in linear logic. *Theor. Comput. Sci.*, 412(20):1884–1902, 2011. `doi:10.1016/j.tcs.2010.12.017`.

[DHR96] Vincent Danos, Hugo Herbelin, and Laurent Regnier. Game semantics & abstract machines. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 394–405. IEEE Computer Society, 1996. `doi:10.1109/LICS.1996.561456`.

[DJ03] Vincent Danos and Jean-Baptiste Joinet. Linear logic and elementary time. *Inf. Comput.*, 183(1):123–137, 2003. `doi:10.1016/S0890-5401(03)00010-5`.

[DLM12] Ugo Dal Lago and Simone Martini. On constructor rewrite systems and the lambda calculus. *Logical Methods in Computer Science*, 8(3), 2012. `doi:10.2168/LMCS-8(3:12)2012`.

[DR04]     Vincent Danos and Laurent Regnier. Head linear reduction. Technical report, 2004.

[EG16]     Thomas Ehrhard and Giulio Guerrieri. The bang calculus: an untyped lambda-calculus generalizing call-by-name and call-by-value. In James Cheney and Germán Vidal, editors, *Proceedings of the 18th International Symposium on Principles and Practice of Declarative Programming, Edinburgh, United Kingdom, September 5-7, 2016*, pages 174–187. ACM, 2016. `doi:10.1145/2967973.2968608`.

[Ehr14]    Thomas Ehrhard. A new correctness criterion for MLL proof nets. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 38:1–38:10. ACM, 2014. `doi:10.1145/2603088.2603125`.

[ER06]     Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theor. Comput. Sci.*, 364(2):166–195, 2006. `doi:10.1016/j.tcs.2006.08.003`.

[FM99]     Maribel Fernández and Ian Mackie. A calculus for interaction nets. In Gopalan Nadathur, editor, *Principles and Practice of Declarative Programming, International Conference PPDP'99, Paris, France, September 29 - October 1, 1999, Proceedings*, volume 1702 of *Lecture Notes in Computer Science*, pages 170–187. Springer, 1999. `doi:10.1007/10704567\_10`.

[FR19]     Claudia Faggian and Simona Ronchi Della Rocca. Lambda calculus and probabilistic computation. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE, 2019. `doi:10.1109/LICS.2019.8785699`.

[FS14]     Maribel Fernández and Nikolaos Siafakas. Labelled calculi of resources. *J. Log. Comput.*, 24(3):591–613, 2014. `doi:10.1093/logcom/exs021`.

[GdPR00]   Neil Ghani, Valeria de Paiva, and Eike Ritter. Linear explicit substitutions. *Log. J. IGPL*, 8(1):7–31, 2000. `doi:10.1093/jigpal/8.1.7`.

[Gir87]    Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987. `doi:10.1016/0304-3975(87)90045-4`.

[Gir98]    Jean-Yves Girard. Light linear logic. *Inf. Comput.*, 143(2):175–204, 1998. `doi:10.1006/inco.1998.2700`.

[GSS92]    Jean-Yves Girard, Andre Scedrov, and Philip J. Scott. Bounded linear logic: A modular approach to polynomial-time computability. *Theor. Comput. Sci.*, 97(1):1–66, 1992. `doi:10.1016/0304-3975(92)90386-T`.

[Hin64]    J.R. Hindley. *The Church-Rosser Property and a Result in Combinatory Logic*. PhD thesis, University of Newcastle-upon-Tyne, 1964.

[Joi77]    Jean-Baptiste Joinet. *Étude de la normalisation du calcul des séquents*. Phd thesis, Université Paris 7, 1977.

[Kes07]    Delia Kesner. The theory of calculi with explicit substitutions revisited. In Jacques Duparc and Thomas A. Henzinger, editors, *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11-15, 2007, Proceedings*, volume 4646 of *Lecture Notes in Computer Science*, pages 238–252. Springer, 2007. `doi:10.1007/978-3-540-74915-8\_20`.

[Kes09]    Delia Kesner. A theory of explicit substitutions with safe and full composition. *Log. Methods Comput. Sci.*, 5(3), 2009. URL: `http://arxiv.org/abs/0905.2539`.

[Kes22]    Delia Kesner. A fine-grained computational interpretation of girard's intuitionistic proof-nets. *Proc. ACM Program. Lang.*, 6(POPL):1–28, 2022. `doi:10.1145/3498669`.

[KL07]     Delia Kesner and Stéphane Lengrand. Resource operators for lambda-calculus. *Inf. Comput.*, 205(4):419–473, 2007. `doi:10.1016/j.ic.2006.08.008`.

[Laf04]    Yves Lafont. Soft linear logic and polynomial time. *Theor. Comput. Sci.*, 318(1-2):163–180, 2004. `doi:10.1016/j.tcs.2003.10.018`.

[LM92]     Patrick Lincoln and John C. Mitchell. Operational aspects of linear lambda calculus. In *Proceedings of the Seventh Annual Symposium on Logic in Computer Science (LICS '92), Santa Cruz, California, USA, June 22-25, 1992*, pages 235–246. IEEE Computer Society, 1992. `doi:10.1109/LICS.1992.185536`.

[LTdF04]  Olivier Laurent and Lorenzo Tortora de Falco. Slicing polarized additive normalization. In *Linear Logic in Computer Science*, volume 316 of *London Mathematical Society Lecture Note Series*, pages 247–282. Cambridge University Press, November 2004.

[Maz17]  Damiano Mazza. *Polyadic Approximations in Logic and Computation. Habilitation* thesis, Université Paris 13, 2017.

[Mel95]  Paul-André Melliès. Typed lambda-calculi with explicit substitutions may not terminate. In Mariangiola Dezani-Ciancaglini and Gordon D. Plotkin, editors, *Typed Lambda Calculi and Applications, Second International Conference on Typed Lambda Calculi and Applications, TLCA '95, Edinburgh, UK, April 10-12, 1995, Proceedings*, volume 902 of *Lecture Notes in Computer Science*, pages 328–334. Springer, 1995. `doi:10.1007/BFb0014062`.

[Mil06]  Robin Milner. Local bigraphs and confluence: Two conjectures: (extended abstract). In Roberto M. Amadio and Iain Phillips, editors, *Proceedings of the 13th International Workshop on Expressiveness in Concurrency, EXPRESS 2006, Bonn, Germany, August 26, 2006*, volume 175 of *Electronic Notes in Theoretical Computer Science*, pages 65–73. Elsevier, 2006. `doi:10.1016/j.entcs.2006.07.035`.

[MOTW99]  John Maraist, Martin Odersky, David N. Turner, and Philip Wadler. Call-by-name, call-by-value, call-by-need and the linear lambda calculus. *Theor. Comput. Sci.*, 228(1-2):175–210, 1999. `doi:10.1016/S0304-3975(98)00358-2`.

[MP94]  Gianfranco Mascari and Marco Pedicini. Head linear reduction and pure proof net extraction. *Theor. Comput. Sci.*, 135(1):111–137, 1994. `doi:10.1016/0304-3975(94)90263-1`.

[MS08]  Ian Mackie and Shinya Sato. A calculus for interaction nets based on the linear chemical abstract machine. *Electron. Notes Theor. Comput. Sci.*, 192(3):59–70, 2008. `doi:10.1016/j.entcs.2008.10.027`.

[MT03]  Harry G. Mairson and Kazushige Terui. On the computational complexity of cut-elimination in linear logic. In Carlo Blundo and Cosimo Laneve, editors, *Theoretical Computer Science, 8th Italian Conference, ICTCS 2003, Bertinoro, Italy, October 13-15, 2003, Proceedings*, volume 2841 of *Lecture Notes in Computer Science*, pages 23–36. Springer, 2003. `doi:10.1007/978-3-540-45208-9\_4`.

[OH06]  Yo Ohta and Masahito Hasegawa. A terminating and confluent linear lambda calculus. In Frank Pfenning, editor, *Term Rewriting and Applications, 17th International Conference, RTA 2006, Seattle, WA, USA, August 12-14, 2006, Proceedings*, volume 4098 of *Lecture Notes in Computer Science*, pages 166–180. Springer, 2006. `doi:10.1007/11805618\_13`.

[PCPT14]  Jorge A. Pérez, Luís Caires, Frank Pfenning, and Bernardo Toninho. Linear logical relations and observational equivalences for session-based concurrency. *Inf. Comput.*, 239:254–302, 2014. `doi:10.1016/j.ic.2014.08.001`.

[PTdF10]  Michele Pagani and Lorenzo Tortora de Falco. Strong normalization property for second order linear logic. *Theor. Comput. Sci.*, 411(2):410–444, 2010. `doi:10.1016/j.tcs.2009.07.053`.

[Reg92]  Laurent Regnier. *Lambda-calcul et réseaux*. PhD thesis, Univ. Paris VII, 1992.

[Rév92]  György E. Révész. A list-oriented extension of the lambda-calculus satisfying the Church-Rosser theorem. *Theor. Comput. Sci.*, 93(1):75–89, 1992. `doi:10.1016/0304-3975(92)90212-X`.

[Rib08]  Colin Riba. Toward a General Rewriting-Based Framework for Reducibility. Technical report, December 2008. URL: `https://hal.archives-ouvertes.fr/hal-00779623`.

[RR97]  Simona Ronchi Della Rocca and Luca Roversi. Lambda calculus and intuitionistic linear logic. *Stud Logica*, 59(3):417–448, 1997. `doi:10.1023/A:1005092630115`.

[RV09]  Luca Roversi and Luca Vercelli. A local criterion for polynomial-time stratified computations. In Marko C. J. D. van Eekelen and Olha Shkaravska, editors, *Foundational and Practical Aspects of Resource Analysis - First International Workshop, FOPARA 2009, Eindhoven, The Netherlands, November 6, 2009, Revised Selected Papers*, volume 6324 of *Lecture Notes in Computer Science*, pages 114–130. Springer, 2009. `doi:10.1007/978-3-642-15331-0\_8`.

[Sau08]  Alexis Saurin. On the relations between the syntactic theories of lambda-mu-calculi. In Michael Kaminski and Simone Martini, editors, *Computer Science Logic, 22nd International Workshop, CSL 2008, 17th Annual Conference of the EACSL, Bertinoro, Italy, September 16-19, 2008. Proceedings*, volume 5213 of *Lecture Notes in Computer Science*, pages 154–168. Springer, 2008. `doi:10.1007/978-3-540-87531-4\_13`.

[SGM02]   David Sands, Jörgen Gustavsson, and Andrew Moran. Lambda calculi and linear speedups. In Torben Æ. Mogensen, David A. Schmidt, and Ivan Hal Sudborough, editors, *The Essence of Computation, Complexity, Analysis, Transformation. Essays Dedicated to Neil D. Jones [on occasion of his 60th birthday]*, volume 2566 of *Lecture Notes in Computer Science*, pages 60–84. Springer, 2002. `doi:10.1007/3-540-36377-7\_4`.

[Sim05]    Alex K. Simpson. Reduction in a linear lambda-calculus with applications to operational semantics. In Jürgen Giesl, editor, *Term Rewriting and Applications, 16th International Conference, RTA 2005, Nara, Japan, April 19-21, 2005, Proceedings*, volume 3467 of *Lecture Notes in Computer Science*, pages 219–234. Springer, 2005. `doi:10.1007/978-3-540-32033-3\_17`.

[Tra09]    Paolo Tranquilli. Confluence of pure differential nets with promotion. In Erich Grädel and Reinhard Kahle, editors, *Computer Science Logic, 23rd international Workshop, CSL 2009, 18th Annual Conference of the EACSL, Coimbra, Portugal, September 7-11, 2009. Proceedings*, volume 5771 of *Lecture Notes in Computer Science*, pages 500–514. Springer, 2009. `doi:10.1007/978-3-642-04027-6\_36`.

[Wad93]    Philip Wadler. A syntax for linear logic. In Stephen D. Brookes, Michael G. Main, Austin Melton, Michael W. Mislove, and David A. Schmidt, editors, *Mathematical Foundations of Programming Semantics, 9th International Conference, New Orleans, LA, USA, April 7-10, 1993, Proceedings*, volume 802 of *Lecture Notes in Computer Science*, pages 513–529. Springer, 1993. `doi:10.1007/3-540-58027-1\_24`.

[Wad12]    Philip Wadler. Propositions as sessions. In Peter Thiemann and Robby Bruce Findler, editors, *ACM SIGPLAN International Conference on Functional Programming, ICFP'12, Copenhagen, Denmark, September 9-15, 2012*, pages 273–286. ACM, 2012. `doi:10.1145/2364527.2364568`.

## Appendix A. Appendix of Sect. 13 (Typed Strong Normalization)

**Theorem 13.12** (Adequacy). *Let $\pi \rhd \Gamma \vdash t : A$ a type derivation. Then $\pi$ is reducible.*

*Proof.* Here we give the cases that are omitted from the body of the paper. Let $\Gamma = x_1 : B_1, .., x_k : B_k$. Cases of the last rule of $\pi$:

- *Multiplicative axiom*:

$$\frac{A \neq \,!B}{m : A \vdash m : A} \ \mathtt{ax_m}$$

  We need to show that $E\langle\!\langle [v_\mathtt{m} {\to} m] m \rangle\!\rangle \in SN_{\to_\mathtt{ss}}$ for $v_\mathtt{m} \in \mathcal{S}_{\llbracket A \rrbracket} \subseteq SN_{\to_\mathtt{ss}}$ and $E \in \llbracket A \rrbracket^\perp$. Note that $E\langle\!\langle [v_\mathtt{m} {\to} m] m \rangle\!\rangle \to_{\mathtt{axm_1}} E\langle\!\langle \{v_\mathtt{m} {\to} m\} m \rangle\!\rangle = E\langle\!\langle v_\mathtt{m} \rangle\!\rangle$ which is in $SN_{\to_\mathtt{ss}}$ by duality, and that $E\langle\!\langle v_\mathtt{m} \rangle\!\rangle = E\langle\!\langle \{v_\mathtt{m} {\to} m\} m \rangle\!\rangle = \{v_\mathtt{m} {\to} m\} E\langle\!\langle m \rangle\!\rangle$. By root cut expansion, $[v_\mathtt{m} {\to} m] E\langle\!\langle m \rangle\!\rangle \in SN_{\to_\mathtt{ss}}$. By structural stability, $[v_\mathtt{m} {\to} m] E\langle\!\langle m \rangle\!\rangle \equiv_\mathtt{cut} E\langle\!\langle [v_\mathtt{m} {\to} m] m \rangle\!\rangle \in SN_{\to_\mathtt{ss}}$.

- *Exchange*:

$$\frac{\rho \ \rhd \Delta, x : B, y : C, \Pi \vdash t : A}{\Delta, y : C, x : B, \Pi \vdash t : A} \ \mathtt{exch}$$

  with $\Gamma = \Delta, y : C, x : B, \Pi$. Let $\Delta = x_1 : B_1, .., x_k : B_k$ and $\Pi = y_1 : C_1, .., y_h : C_h$. We have to show that

$$t' := E\langle\!\langle [v_i {\to} x_i]_\Delta [v {\to} y][v' {\to} x][v'_j {\to} y_j]_\Pi t \rangle\!\rangle \in SN_{\to_\mathtt{ss}}.$$

  for every $v \in \mathcal{S}_{\llbracket C \rrbracket}$, $v' \in \mathcal{S}_{\llbracket B \rrbracket}$, $v_i \in \mathcal{S}_{\llbracket B_i \rrbracket}$ for $i \in \{1, \ldots, k\}$, every $v'_j \in \mathcal{S}_{\llbracket C_j \rrbracket}$ for $i \in \{1, \ldots, h\}$, and every $E \in \llbracket A \rrbracket^\perp$. By *i.h.* on $\rho$, we have

$$t'' := E\langle\!\langle [v_i {\to} x_i]_\Delta [v' {\to} x][v {\to} y][v'_j {\to} y_j]_\Pi t \rangle\!\rangle \in SN_{\to_\mathtt{ss}}.$$

  By structural stability and the independence of cuts in the definition of reducibility, $t'' \equiv_\mathtt{cut} t' \in SN_{\to_\mathtt{ss}}$.

- *Cut*:

$$\frac{\pi_l \ \rhd \Gamma_l \vdash L\langle v \rangle : B \qquad \pi_s \ \rhd \Gamma_s, x : B \vdash s : A \qquad \Gamma_l \# (\Gamma_s, x : B)}{\Gamma_l, \Gamma_s \vdash L\langle [v {\to} x] s \rangle : A} \ \mathtt{cut}$$

  with $t = L\langle [v {\to} x] s \rangle$ and $\Gamma = \Gamma_l, \Gamma_s$. We have to show that

$$t' := E\langle\!\langle [v_i {\to} x_i]_\Gamma L\langle [v {\to} x] s \rangle \rangle\!\rangle \in SN_{\to_\mathtt{ss}}.$$

  for every $v_i \in \mathcal{S}_{\llbracket B_i \rrbracket}$ for $i \in \{1, \ldots, k\}$ and every $E \in \llbracket A \rrbracket^\perp$. By *i.h.* on $\pi_l$, we have $[v_i {\to} x_i]_{\Gamma_l} L\langle v \rangle \in \llbracket B \rrbracket$. By *i.h.*, $\pi_s$ is reducible, and by the extended reducibility clause (Lemma 13.11) we obtain $[v_i {\to} x_i]_{\Gamma_s} [v_i {\to} x_i]_{\Gamma_l} L\langle [v {\to} x] s \rangle \in \llbracket A \rrbracket$, that is,

$$E\langle\!\langle [v_i {\to} x_i]_{\Gamma_s} [v_i {\to} x_i]_{\Gamma_l} L\langle [v {\to} x] s \rangle \rangle\!\rangle \in SN_{\to_\mathtt{ss}}.$$

  By structural stability and the independence of cuts in the definition of reducibility,

$$E\langle\!\langle [v_i {\to} x_i]_{\Gamma_s} [v_i {\to} x_i]_{\Gamma_l} L\langle [v {\to} x] s \rangle \rangle\!\rangle \equiv_\mathtt{cut} t' \in SN_{\to_\mathtt{ss}}.$$

- *Exponential inductive cases.*
  - *Weakening*:

$$\frac{\rho \ \rhd \Delta \vdash t : A \qquad e \text{ fresh}}{\Delta, e :\, !B \vdash t : A} \ \mathtt{w}$$

    with $\Gamma = \Delta, e :\, !B$. We use the notation $x_i : B_i$ for the assignments in $\Delta$. We have to show that $E\langle\!\langle [v_i {\to} x_i]_\Delta [!u {\to} e] t \rangle\!\rangle \in SN_{\to_\mathtt{ss}}$ for every $u \in \llbracket B \rrbracket$, every $v_i \in \mathcal{S}_{\llbracket B_i \rrbracket}$ for $i \in \{1, \ldots, k\}$, and every $E \in \llbracket A \rrbracket^\perp$. By *i.h.* on $\rho$, $E\langle\!\langle [v_i {\to} x_i]_\Delta t \rangle\!\rangle \in SN_{\to_\mathtt{ss}}$. By freshness of $e$,

$$E\langle\!\langle [v_i {\to} x_i]_\Delta t \rangle\!\rangle = E\langle\!\langle \{!u {\to} e\}[v_i {\to} x_i]_\Delta t \rangle\!\rangle = \{!u {\to} e\} E\langle\!\langle [v_i {\to} x_i]_\Delta t \rangle\!\rangle.$$

Since $u \in [\![B]\!]$, we have $u \in SN_{\to_{ss}}$ by the properties of candidates (Prop. 13.9), thus $!u \in SN_{\to_{ss}}$ by extension. By root cut expansion, $[!u{\to}e]E\langle\!\langle[v_i{\to}x_i]_\Delta t\rangle\!\rangle \in SN_{\to_{ss}}$. By structural stability and the independence of cuts in the definition of reducibility, $[!u{\to}e]E\langle\!\langle[v_i{\to}x_i]_\Delta t\rangle\!\rangle \equiv_{\texttt{cut}} E\langle\!\langle[v_i{\to}x_i]_\Delta[!u{\to}e]t\rangle\!\rangle \in SN_{\to_{ss}}$.

– *Dereliction*:
$$\frac{\rho \rhd \Delta, x : B \vdash s : A \qquad e \text{ fresh}}{\Delta, e : !B \vdash (\!|e?x|\!)s : A} \, !_l$$

with $t = (\!|e?x|\!)s$ and $\Gamma = \Delta, e : !B$. We use the notation $x_i : B_i$ for the assignments in $\Delta$. We have to show that

$$E\langle\!\langle[v_i{\to}x_i]_\Delta[!u{\to}e](\!|e?x|\!)s\rangle\!\rangle \in SN_{\to_{ss}}$$

for every $u \in [\![B]\!]$, every $v_i \in \mathcal{S}_{[\![B_i]\!]}$ for $i \in \{1,\ldots,k\}$, and every $E \in [\![A]\!]^\perp$. Let $u = L\langle v\rangle$. By *i.h.*, $\rho$ is reducible. by the extended reducibility clause (Lemma 13.11), $[v_i{\to}x_i]_\Delta L\langle[v{\to}x]s\rangle \in [\![A]\!]$, that is,

$$E\langle\!\langle[v_i{\to}x_i]_\Delta L\langle[v{\to}x]s\rangle\rangle\!\rangle \in SN_{\to_{ss}}.$$

Note that by freshness of $e$ and the independence of cuts in the definition of reducibility, we have

$$E\langle\!\langle[v_i{\to}x_i]_\Delta L\langle[v{\to}x]s\rangle\rangle\!\rangle = E\langle\!\langle[v_i{\to}x_i]_\Delta\{!u{\to}e\}(\!|e?x|\!)s\rangle\!\rangle = \{!u{\to}e\}E\langle\!\langle[v_i{\to}x_i]_\Delta(\!|e?x|\!)s\rangle\!\rangle.$$

Since $u \in [\![B]\!]$, we have $u \in SN_{\to_{ss}}$ by the properties of candidates (Prop. 13.9), thus $!u \in SN_{\to_{ss}}$ by extension. By root cut expansion, $[!u{\to}e]E\langle\!\langle[v_i{\to}x_i]_\Delta(\!|e?x|\!)s\rangle\!\rangle \in SN_{\to_{ss}}$. By structural stability,

$$[!u{\to}e]E\langle\!\langle[v_i{\to}x_i]_\Delta(\!|e?x|\!)s\rangle\!\rangle \equiv_{\texttt{cut}} E\langle\!\langle[v_i{\to}x_i]_\Delta[!u{\to}e](\!|e?x|\!)s\rangle\!\rangle \in SN_{\to_{ss}}.$$

- *Multiplicative inductive cases.*
  – *Tensor*:
$$\frac{\pi_s \rhd \Gamma_s \vdash s : B \qquad \pi_u \rhd \Gamma_u \vdash u : C \qquad \Gamma_s \# \Gamma_u}{\Gamma_s, \Gamma_u \vdash (s, u) : B \otimes C} \, \otimes_r$$
with $t = (s, u)$, $A = B \otimes C$, and $\Gamma = \Gamma_s, \Gamma_u$. We have to show that $t' := E\langle\!\langle[v_i{\to}x_i]_\Gamma(s, u)\rangle\!\rangle$ is $SN_{\to_{ss}}$ for every $v_i \in \mathcal{S}_{[\![B_i]\!]}$ for $i \in \{1,\ldots,k\}$ and every $E \in [\![B \otimes C]\!]^\perp$. By *i.h.* on $\pi_s$ and $\pi_u$, $s' := [v_i{\to}x_i]_{\Gamma_s}s \in [\![B]\!]$ and $u' := [v_i{\to}x_i]_{\Gamma_u}u \in [\![C]\!]$, and so $(s', u') \in \mathcal{S}_{[\![B \otimes C]\!]} \subseteq [\![B \otimes C]\!]$, that is, $E\langle\!\langle(s', u')\rangle\!\rangle \in SN_{\to_{ss}}$. By structural stability,

$$\begin{array}{ll}
E\langle\!\langle(s', u')\rangle\!\rangle & = \\
E\langle\!\langle([v_i{\to}x_i]_{\Gamma_s}s, [v_i{\to}x_i]_{\Gamma_u}u)\rangle\!\rangle & \equiv_{\texttt{cut}} \\
E\langle\!\langle[v_i{\to}x_i]_{\Gamma_s}[v_i{\to}x_i]_{\Gamma_u}(s, u)\rangle\!\rangle & = \\
t' \in SN_{\to_{ss}}. &
\end{array}$$

.
  – *Par*:
$$\frac{\rho \rhd \Delta, x : B, y : C \vdash s : A \qquad m \text{ fresh}}{\Delta, m : B \otimes C \vdash (\!|m \mathbin{\mathscr{R}} x, y|\!)s : A} \, \otimes_l$$
with $t = (\!|m \mathbin{\mathscr{R}} x, y|\!)s$ and $\Gamma = \Delta, m : B \otimes C$. We use the notation $x_i : B_i$ for the assignments in $\Delta$. The reducibility clause to prove is

$$t' := E\langle\!\langle[v_i{\to}x_i]_\Delta[(u, r){\to}m](\!|m \mathbin{\mathscr{R}} x, y|\!)s\rangle\!\rangle \in SN_{\to_{ss}}$$

for every $v_i \in \mathcal{S}_{[\![B_i]\!]}$ for $i \in \{1,\ldots,k\}$, every $(u, r) \in \mathcal{S}_{[\![B \otimes C]\!]}$, and every $E \in [\![A]\!]^\perp$. Let $u = L\langle v\rangle$ and $r = L'\langle v'\rangle$. By *i.h.*, $\rho$ is reducible, and by the extended reducibility

clause (Lemma 13.11), $[v_i{\to}x_i]_\Delta L'\langle[v{\to}x]L''\langle[v'{\to}y]s\rangle\rangle \in [\![A]\!]$, that is,

$$t'' := E\langle\!\langle[v_i{\to}x_i]_\Delta L'\langle[v{\to}x]L''\langle[v'{\to}y]s\rangle\rangle\rangle\!\rangle \in SN_{\to_{ss}}.$$

By root cut expansion, we have $[(u,r){\to}m]E\langle\!\langle[v_i{\to}x_i]_\Delta(\!|m\,\Im x, y|\!)s\rangle\!\rangle \in SN_{\to_{ss}}$. By structural stability and the independence of cuts in the definition of reducibility,

$$E\langle\!\langle[v_i{\to}x_i]_\Delta[(u,r){\to}m](\!|m\,\Im x, y|\!)s\rangle\!\rangle \in SN_{\to_{ss}}$$

as required. $\square$