

GLUING RESOURCE PROOF-STRUCTURES: INHABITATION AND INVERTING THE TAYLOR EXPANSION

GIULIO GUERRIERI^a, LUC PELLISSIER^b, AND LORENZO TORTORA DE FALCO^c

^a Huawei Research, Edinburgh Research Centre, Edinburgh, United Kingdom
e-mail address: giulio.guerrieri@huawei.com

^b Université Paris Est Créteil, LACL, F-94010 Créteil, France
e-mail address: luc.pellissier@lacl.fr

^c Università Roma Tre, Dipartimento di Matematica e Fisica, Rome, Italy
e-mail address: tortora@uniroma3.it

ABSTRACT. A Multiplicative-Exponential Linear Logic (MELL) proof-structure can be expanded into a set of resource proof-structures: its Taylor expansion. We introduce a new criterion characterizing (and deciding in the finite case) those sets of resource proof-structures that are part of the Taylor expansion of some MELL proof-structure, through a rewriting system acting both on resource and MELL proof-structures. We also prove semi-decidability of the type inhabitation problem for cut-free MELL proof-structures.

1. INTRODUCTION

The Taylor expansion. Girard’s linear logic (LL, [Gir87]) is a refinement of intuitionistic and classical logic that isolates the infinitary parts of reasoning in two dual modalities: the *exponentials* ! and ?. They give a logical status to operations of memory management such as *copying* and *erasing*: a linear (*i.e.* exponential-free) proof corresponds—via the Curry–Howard isomorphism—to a program that uses its argument *linearly*, *i.e.* exactly once, while an exponential proof corresponds to a program that can use its argument *at will*.

The intuition that linear programs are analogous to linear functions (as studied in linear algebra) while exponential programs mirror a more general class of analytic functions got a technical incarnation in Ehrhard’s work [Ehr02, Ehr05] on LL-based denotational semantics for the λ -calculus. This investigation has been then internalized in the syntax, yielding the *resource λ -calculus* [ER03, ER08], inspired by [Bou93]: there, copying and erasing are forbidden and replaced by the possibility to apply a function to a *bag* of resource λ -terms which specifies how many times (in a finite number) an argument can be linearly passed to the function, so as to represent only bounded computations.

The *Taylor expansion* [ER08] (more precisely, its support; but here and throughout the paper we do not consider the rational coefficients in the Taylor expansion) associates with every ordinary λ -term a—generally infinite—set of resource λ -terms, recursively approximating

Key words and phrases: linear logic, Taylor expansion, proof-structure, pullback, natural transformation.

the usual application: the Taylor expansion of the λ -term MN is made up of all resource λ -terms of the form $t[u_1, \dots, u_n]$, where t is a resource λ -term in the Taylor expansion of M , and $[u_1, \dots, u_n]$ is a bag of n (for any $n \geq 0$) resource λ -terms in the Taylor expansion of N . Roughly, the idea is to decompose a program into a set of purely “resource-sensitive programs”, all of them containing only bounded (although possibly non-linear) calls to inputs. The notion of Taylor expansion has had many applications in the theory of the λ -calculus, *e.g.* in the study of linear head reduction [ER06a], normalization [PTV16, Vau17], Böhm trees [BHP13, KMP20, BM20], λ -theories [MR14], intersection types [MPV18]. In general, understanding the relation between a program and its Taylor expansion renews the logical approach to the quantitative analysis of computation started with the inception of LL.

A natural question is the *inverse Taylor expansion problem*: Which sets of resource λ -terms are included in the Taylor expansion of a same λ -term? How to characterize them? Ehrhard and Regnier [ER08] defined a simple *coherence* binary relation such that a finite set of resource λ -terms is included in the Taylor expansion of a λ -term if and only if all the elements of this set are pairwise coherent. Coherence is crucial in many structural properties of the resource λ -calculus, such as in the proof that in the λ -calculus normalization and Taylor expansion commute [ER06a, ER08].

We aim to solve the inverse Taylor expansion problem in the more general context of LL, more precisely in the *multiplicative-exponential fragment* MELL of LL, being aware that for MELL no coherence relation can characterize the solutions (see below). Our characterization is constructive, in that it allows us to define a *decision* procedure to solve the inverse Taylor expansion problem in the *finite* case. A side effect of this investigation is apparently unrelated to the notion of Taylor expansion: we characterize MELL formulas that are inhabited by cut-free MELL proof-structures, so as to prove semi-decidability of the *type inhabitation problem* for cut-free MELL proof-structures (again, see below).

Proof-nets, proof-structures and their Taylor expansion: seeing trees behind graphs. In MELL, linearity and the sharp analysis of computations naturally lead to represent proofs in a more general *graph*-like syntax instead of a term-like or tree-like one.¹ Indeed, linear negation is involutive and classical duality can be interpreted as the possibility of juggling between different conclusions, without a distinguished output [Par92]. Graphs representing proofs in MELL are called *proof-nets*: their syntax is richer and more expressive than the λ -calculus (which corresponds to an intuitionistic implicative fragment of MELL). Contrary to λ -terms, proof-nets are special inhabitants of the wider land of *proof-structures*. A proof-structure is any “graph” that can be build in the language of proof-nets and it need not represent a proof in MELL. Proof-nets can be characterized, among proof-structures, by abstract (geometric) conditions called correctness criteria [Gir87].

Proof-structures are well-behaved for performing computations: indeed, cut-elimination steps can be defined for proof-structures, and proof-nets can also be seen as the proof-structures with a good behavior with respect to cut-elimination [Béc98]. Furthermore, proof-structures can be interpreted in denotational models and proof-nets can be characterized among them by semantic means [Ret97]. It is then natural to attack problems in the general framework of proof-structures. In our work, correctness plays no role at all, hence we will consider proof-structures and not only proof-nets. MELL proof-structures are a particular kind of graphs, whose edges are labeled by MELL formulas and vertices (aka cells) by MELL

¹A term-like object is essentially a tree, with one output (its root) and many inputs (its other leaves).

connectives, and for which special subgraphs are highlighted, the *boxes*, representing the parts of the proof-structure that can be discarded and copied (*i.e.* called an unbounded number of times) during cut-elimination. A box is delimited from the rest of a proof-structure by exponential modalities: its border is made of one $!$ -cell, its principal door, and arbitrarily many $?$ -cells, its auxiliary doors. Boxes are either nested or disjoint (they cannot partially overlap), so as to add a tree-like structure to proof-structures *aside* from their graph-like nature.

As in the λ -calculus, one can define box-free *resource* (or DiLL_0) *proof-structures*²[ER06b], where $!$ -cells make resources available boundedly, and the *Taylor expansion* of MELL proof-structures into these resource proof-structures, that recursively copies the content of the boxes an arbitrary number of times. In fact, as somehow anticipated by Boudes [Bou09], such a Taylor expansion operation can be carried on any tree-like structure. This primitive, abstract, notion of Taylor expansion can then be pulled back to the structure of interest (in this case, the resource proof-structures), as shown in [GPT19] and put forth again here.

The question of coherence for proof-structures. The *inverse Taylor expansion problem* has a natural counterpart for MELL proof-structures: given a set Π of resource proof-structures, is there a MELL proof-structure the Taylor expansion of which includes Π ? Pagani and Tasson [PT09] give the following answer: it is possible to decide whether a finite set of resource proof-structures is a subset of the Taylor expansion of a same MELL proof-structure (and even possible to do it in nondeterministic polynomial time); but unlike the λ -calculus, the structure of the relation “being part of the Taylor expansion of a same proof-structure” is *much more* complicated than a binary (or even n -ary) coherence. Indeed, for any $n > 1$, it is possible to find $n + 1$ resource proof-structures such that any n of them are in the Taylor expansion of some MELL proof-structure, but there is no MELL proof-structure whose Taylor expansion has all the $n+1$ as elements (see our Example 8.6 and [Tas09, pp. 244-246]).

In this work, we introduce a new combinatorial criterion, *gluability*, for deciding whether a set of resource proof-structures is a subset of the Taylor expansion of some MELL proof-structure, based on a *rewriting system* on lists of MELL formulas. Our criterion is more general and simpler than the one of [PT09], which is limited to the *cut-free* case with *atomic axioms* and characterizes only *finite* sets: we do not have these limitations. Akin to [PT09], our criterion yields a *decision procedure* for the inverse Taylor expansion problem when the set of resource proof-structures given as input is *finite*. We believe that our criterion is a useful tool for studying proof-structures. We conjecture that it can be used to show that a binary coherence relation exists for resource proof-structures satisfying a suitable geometric restriction. It might also shed light on correctness and sequentialization.

As the proof-structures we consider are typed, an unrelated difficulty arises: a resource proof-structure ρ of type A might not be in the Taylor expansion of any cut-free MELL proof-structure, not because it does not respect the structure imposed by the Taylor expansion, but because there is no cut-free MELL proof-structure of type A , and ρ can “mask” this “untypeability”.³ To solve this issue, we enrich the resource (but not MELL) proof-structure syntax with a “universal” proof-structure: a special \boxtimes -cell (*daimon*) that can have any number of outputs of any types, representing information plainly missing (see Section 11 for more details and the way this matter is handled by Pagani and Tasson [PT09]).

²Aka differential proof-structures [dC16], differential nets [ER06b, MP07, dC18], simple nets [PT09].

³Similarly, in the λ -calculus, there is no closed λ -term of type $X \rightarrow Y$ with $X \neq Y$ atomic, but the resource λ -term $(\lambda f.f)[\]$ can be given that type: the empty bag $[\]$ kills any information on the argument.

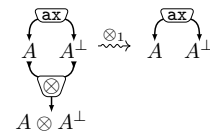
Our contribution. This paper is the long version of [GPT20] and we keep its structure and main results: the *gluability* criterion that solves the inverse Taylor expansion problem in MELL proof-structures (Theorem 8.3), and the way we prove it, which consists in showing that the Taylor expansion defines a *natural transformation* (Theorem 7.5) from the realm of resource proof-structures to the realm of MELL proof-structures (see Section 2 for an informal explanation). With respect to [GPT20], the main novelties are:

- (1) Following [GPT19], we introduce rigorous definitions of all the notions of graph theory (Section 3) involved in the definition of proof-structures and Taylor expansion. In this way, we can present here a purely graphical definition of MELL proof-structures (Section 4), so as to keep Girard’s original intuition of a proof-structure as a graph even in MELL and to avoid *ad hoc* technicalities to identify the border and the content of a box. This is not only an aesthetic issue but also practical, in that our MELL proof-structures are manageable: sophisticated operations on them can be easily defined. For instance, we give an elegant definition of their Taylor expansion by means of *pullbacks* (Section 5).
- (2) With respect to [GPT20], here we use daimons in a different and more limited way. In particular, unlike [GPT20], our MELL proof-structures do not contain any \bowtie -cell. Thus, our results refer to a more standard and interesting definition of MELL proof-structures, and we deal with less syntactic categories than in [GPT20], simplifying the presentation and fixing some technical inaccuracies in a few definitions and lemmas in [GPT20].
- (3) Consequently, our results (notably, naturality and gluability) are more informative, and allow us to *decide* the inverse Taylor expansion problem in the *finite* case (Theorem 9.9).
- (4) We solve the apparently unrelated *type inhabitation problem* for cut-free MELL proof-structures (Theorem 8.10), not considered in [GPT20]: our rewrite system *semi-decides* if, for a list Γ of MELL formulas, there is a cut-free MELL proof-structure of type Γ . We only provide a semi-algorithm: we can guess a rewriting and check its correctness; but there is no bound on its length. Inhabitation problems are well-studied in many type systems for the λ -calculus, but no results are in the literature for MELL proof-structures. Our contribution may shed some light on the open problem of decidability of MELL.
- (5) Akin to [GPT20], to simplify the presentation, we first focus on proof-structures restricted to atomic axioms, then Section 10 shows how to lift our results to the non-atomic case. Compared to [GPT20], the lift is more elegant and requires less *ad hoc* adjustments.

2. OUTLINE AND TECHNICAL ISSUES

Rewriting. The essence of our rewrite system is not located in proof-structures but in lists of MELL formulas (Definition 6.1). Roughly, the rewrite system is generated by elementary steps akin to rules of sequent calculus read from the *bottom up*: they act on a list of conclusions, analogous to a monolaterous right-handed sequent. These steps can be seen as morphisms in a category **Sched** whose objects are lists of MELL formulas, and are actually more sequentialized than sequent calculus rules, as they do not allow for commutation. For instance, the rule corresponding to the introduction of a \otimes on the i^{th} formula, is defined as $\otimes_i : (C_1, \dots, C_{i-1}, A \otimes B, C_{i+1}, \dots, C_n) \rightarrow (C_1, \dots, C_{i-1}, A, B, C_{i+1}, \dots, C_n)$.

These rewrite steps then act on MELL proof-structures, coherently with their type, by modifying (most of the times, erasing) a cell immediately above the conclusion of the proof-structure. Formally, this means that there is a functor \mathbf{qMELL} from **Sched** to the category **Rel** of sets



and relations, associating with every list of MELL formulas the set of MELL proof-structures with these conclusions, and with every rewrite step a relation implementing it (Definition 6.4). The rules *deconstruct* the proof-structure, starting from its conclusions. The rule \otimes_1 acts by removing a \otimes -cell on the first conclusion, replacing it by two conclusions.

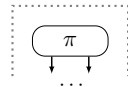
These rules can only act on specific proof-structures, and indeed, capture a lot of their structure: \otimes_i can be applied to a MELL proof-structure R if and only if R has a \otimes -cell in the conclusion i (as opposed to, say, an axiom). So, in particular, every proof-structure is completely characterized by any sequence rewriting it to the empty proof-structure.

Naturality. The same rules also act on sets of resource (aka DiLL₀) proof-structures, defining the functor \mathfrak{PqDiLL}_0 from the category **Sched** of rewrite steps into the category **Rel** (Definition 7.4). When carefully defined, the Taylor expansion induces a *natural transformation* from \mathfrak{PqDiLL}_0 to **qMELL** (Theorem 7.5). By applying this naturality repeatedly, we get our characterization (Theorem 8.3): a set of resource proof-structures Π is a subset of the Taylor expansion of a MELL proof-structure if and only if Π is *gluable*, that is, there is a sequence rewriting Π to the singleton of the *empty* proof-structure.

The naturality property is not only a mean to obtain our characterization, but also an interesting result in itself: natural transformations can often be used to express fundamental properties in a mathematical context. In this case, the *Taylor expansion is natural* with respect to the possibility of building a (MELL or resource) proof-structure by adding a cell to its conclusions or boxing it. Said differently, naturality of the Taylor expansion roughly means that the rewrite rules that deconstruct a MELL proof-structure R and a set of resource proof-structures in the Taylor expansion of R mimic each other.

Quasi-proof-structures and mix. Our rewrite rules consume proof-structures from their conclusions. The rule corresponding to boxes in MELL opens a box by deleting its principal door (a !-cell) and its border, while for a resource proof-structure it deletes a !-cell and separates the different copies of the content of the box (possibly) represented by such a !-cell. This operation is problematic in a twofold way. In a resource proof-structure, where the border of boxes is not marked, it is not clear how to identify such copies. On the other side, in a MELL proof-structure the content of a box is not to be treated as if it were at the same level as what is outside of the box: it can be copied many times or erased, while what is outside boxes cannot, and treating the content in the same way as the outside suppresses this distinction, which is crucial in LL. So, we need to remember that the content of a box, even if it is at depth 0 (*i.e.* not contained in any other box) after erasing the box wrapping it by means of our rewrite rules, is not to be mixed with the rest of the structure at depth 0.

In order for our proof-structures to provide this information, we need to generalize them and consider that a proof-structure can have a *forest* of boxes, instead of just a tree: this yields the notion of *quasi-proof-structure* (Definition 4.5). In this way, according to our rewrite rules, opening a box by deleting its principal door amounts to taking a box in the tree and disconnecting it from its root, creating a new tree. We draw this in a quasi-proof-structure by surrounding each *component*—made of the objects having the same root—with a dashed line, open from the bottom, remembering the phantom presence of the border of the box, even if it was erased. This allows one to open the box only when it is alone in a component, just surrounded by a dashed line (Definition 6.3).



This is not merely a technical remark, as this generalization gives a status to the `mix` rule of LL: indeed, mixing two proofs amounts to taking two proofs and considering them as one, without any other modifications. Here, it amounts to taking two proof-structures, each with its box-tree, and considering them as one by merging the roots of their trees (see the `mix` step in Definition 6.3). We embed this design decision up to the level of formulas, which are segregated in different zones that have to be mixed before interacting. Indeed, our rewrite rules actually act on conclusions arranged as a *list of lists* of MELL formulas.

Geometric invariance and emptiness: the filled Taylor expansion. The use of forests instead of trees for the nesting structure of boxes, where the different roots are thought of as the contents of long-gone boxes, has an interesting consequence in the Taylor expansion: indeed, an element of the Taylor expansion of a proof-structure contains an arbitrary number of copies of the contents of the boxes, in particular *zero*. If we think of the part at depth 0 of a MELL proof-structure as inside an invisible box, its content can be deleted in some elements of the Taylor expansion just as any other box.⁴ As erasing completely the conclusions would cause the Taylor expansion not preserve the conclusions (which would lead to technical complications), we introduce the *filled Taylor expansion* (Definition 5.8), which contains not only the elements of the usual Taylor expansion, but also elements of the Taylor expansion where one component has been erased and replaced by a \boxtimes -cell (*daimon*), representing lack of information, apart from the number and types of the conclusions. Roughly, a \boxtimes -cell is a placeholder for any DiLL₀ proof-structure of given conclusions.

Gluability and cut-free gluability. Our gluability criterion is based on local rewritings, which yields a geometric and modular approach to the inverse Taylor expansion problem. From the geometric point of view, there is nothing special in the elementary rewrite step corresponding to the cut rule, and it is natural to prove our gluability criterion in presence of cuts (Theorem 8.3.1). But from the proof-theoretical point of view, the gulf separating cut-free proof-structures from the others shows up: for every MELL formula A , the set of MELL proof-structures with conclusion of type A is never empty, while this might very well be the case if we restrict to cut-free MELL proof-structures (see Example 8.5 and Remark 8.8). The modularity of our proofs allows to straightforwardly adapt the criterion to the cut-free case (Theorem 8.3.2) and thus to state correctly (and easily solve) the *type inhabitation problem* for (cut-free) MELL proof-structures (Theorem 8.10).

Outline. Section 3 recalls some preliminary notions on graph theory. In Section 4 we define (MELL and DiLL₀) proof-structures and quasi-proof-structures with atomic axioms. Section 5 defines the notion of Taylor expansion. Section 6 introduces the rewriting rules on lists of lists of formulas and lifts them to MELL quasi-proof-structures via the functor \mathfrak{qMELL} . In Section 7 we lift the rewriting rules to DiLL₀ quasi-proof-structures via the functor \mathfrak{PqDiLL}_0 and we show our first main result: the Taylor expansion induces a natural transformation between the two functors. Section 8 proves two other main results: the solution of the inverse Taylor expansion problem (via a gluability criterion) and the solution of the type inhabitation problem in MELL. Section 9 proves that the inverse Taylor expansion problem is decidable in the finite case. In Section 10 we show how to adapt our method when axioms are not necessarily atomic. Section 11 concludes with some final remarks.

⁴The dual case, of copying the contents of a box, poses no problem in our approach.

3. PRELIMINARIES ON GRAPHS

Graphs with half-edges. There are many formalizations of the familiar notion of graph. Here we adopt the one due to [BM07]:⁵ a graph is still a set of edges and a set of vertices, but edges are now split in halves, allowing some of them to be hanging. Splitting every edge in two has at least four features of particular interest to represent LL proof-structures:

- two half-edges are connected by an involution, thus defining an edge linking two vertices (possibly the same vertex). The fixed points of this involution are “hanging” edges, linked to a vertex only on one endpoint: they are well suited for representing the conclusions of a proof-structure. In this way it is also easy to define some intuitive but formally tricky operations such as grafting/substituting a graph into/for another graph (see Example 3.4);
- given any vertex v in a graph τ , it is natural to define the *corolla* of v , that is v itself with the half-edges linked to it; τ is the union of its corollas, glued together by the involution;
- while studying proof-structures, it is often necessary to treat them both as directed and undirected graphs. With this definition of graph, an orientation, a labeling and a coloring, are structures on top of the structure of the undirected graph (see Definition 3.3);
- this definition of graph allows a uniform syntax to represent both proof-structures and other structures of interest (*e.g.*, the box-tree of a proof-structure). In this way, we avoid appealing to *ad hoc* conditions in the definitions of proof-structure and Taylor expansion, which are then more compact and rely only on notions from graph theory.

Definition 3.1 (graph). A (finite)⁶ *graph* is a quadruple $\tau = (F_\tau, V_\tau, \partial_\tau, j_\tau)$, where

- F_τ is a finite set, whose elements are called *flags* of τ ;
- V_τ is a finite set, whose elements are called *vertices* of τ ;
- $\partial_\tau: F_\tau \rightarrow V_\tau$ is a function associating with each flag its *endpoint*;
- $j_\tau: F_\tau \rightarrow F_\tau$ is an involution, *i.e.* $(j_\tau \circ j_\tau)(f) = f$ for every $f \in F_\tau$.

The graph $\tau = (F_\tau, V_\tau, \partial_\tau, j_\tau)$ is *empty* if $V_\tau = \emptyset$.⁷

A *subgraph* of a graph $\tau = (F_\tau, V_\tau, \partial_\tau, j_\tau)$ is a graph $\sigma = (F_\sigma, V_\sigma, \partial_\sigma, j_\sigma)$ where $F_\sigma \subseteq F_\tau$, $V_\sigma \subseteq V_\tau$, $\partial_\sigma = \partial_\tau \upharpoonright_{F_\sigma}$, and, for all $f \in F_\sigma$, $j_\sigma(f) = j_\tau(f)$ if $j_\tau(f) \in F_\sigma$, otherwise $j_\sigma(f) = f$.

In a graph $\tau = (F_\tau, V_\tau, \partial_\tau, j_\tau)$, a *tail* of τ is a fixed point of the involution j_τ , *i.e.* a flag $f = j_\tau(f)$. A set $\{f, f'\}$ of two flags with $j_\tau(f) = f' \neq f$ is an *edge* of τ between vertices $\partial_\tau(f)$ and $\partial_\tau(f')$; f, f' are the *halves* of the edge; if $\partial_\tau(f) = \partial_\tau(f')$ then the edge is a *loop*.

Given two graphs τ and τ' , it is always possible to consider their disjoint union $\tau \sqcup \tau'$ defined as the disjoint union of the underlying sets and functions.

A one-vertex graph with set of flags F and involution the identity function id_F on F is called a *corolla* (the endpoint of each flag is the only vertex); it is usually denoted by $*_F$.

Given a graph $\tau = (F_\tau, V_\tau, \partial_\tau, j_\tau)$, a vertex $v \in V_\tau$ defines a corolla $\tau_v = (F_v, \{v\}, \partial_\tau \upharpoonright_{F_v}, \text{id}_{F_v})$ where $F_v = \partial_\tau^{-1}(v)$. A graph τ can be seen as the disjoint union of the corollas of its vertices, and the involution gluing some flags (the ones that are not tails in τ) in edges.

⁵The folklore attributes the definition of graphs with half-edges to Kontsevitch and Manin, but the idea can actually be traced back to Grothendieck’s *dessins d’enfant*.

⁶The finiteness condition on F_τ and V_τ can be dropped, so as to allow for possibly infinite graphs. We require it because we only deal with finite graphs.

⁷This implies that ∂_τ is the empty function and $F_\tau = \emptyset$ (since F_τ is the domain of ∂_τ).

Definition 3.2 (graph morphism and isomorphism). Let τ, σ be two graphs. A *graph morphism* $h: \tau \rightarrow \sigma$ from τ to σ is a couple of functions $(h_F: F_\tau \rightarrow F_\sigma, h_V: V_\tau \rightarrow V_\sigma)$ such that $h_V \circ \partial_\tau = \partial_\sigma \circ h_F$ and $h_F \circ j_\tau = j_\sigma \circ h_V$ (h_F and h_V are the *components* of h).

A graph morphism is *empty* (resp. an *identity*) if its components are empty (resp. identity) functions. A *graph isomorphism* is a graph morphism whose components are bijective.

Intuitively, a graph morphism preserves tails and edges. The category **Graph** has graphs as objects and graph morphisms as arrows: indeed, graph morphisms compose in a associative way (by composing their components) and identity graph morphisms are neutral for such a composition. **Graph** is a monoidal category, with disjoint union as a monoidal product.

Graphs with structure. Some structures can be put on top of a graph.

Definition 3.3 (structured graph). Let $\tau = (F_\tau, V_\tau, \partial_\tau, j_\tau)$ be a graph.

- A *labeled graph* (τ, ℓ_τ) with labels in L is a graph τ and a function $\ell_\tau: V_\tau \rightarrow L$.
- A *colored graph* (τ, \mathbf{c}_τ) with colors in C is a graph τ and a function $\mathbf{c}_\tau: F_\tau \rightarrow C$ such that $\mathbf{c}_\tau(f) = \mathbf{c}_\tau(f')$ for the two halves f, f' of any edge of τ .
- A *directed graph* (τ, \mathbf{o}_τ) is a graph τ and a function $\mathbf{o}_\tau: F_\tau \rightarrow \{\mathbf{in}, \mathbf{out}\}$ such that $\mathbf{o}_\tau(f) \neq \mathbf{o}_\tau(f')$ for the two halves f, f' of any edge of τ . If $\mathbf{o}_\tau(f) = \mathbf{out}$ and $\mathbf{o}_\tau(f') = \mathbf{in}$, $\{f, f'\}$ is said an *edge* of τ from $\partial_\tau(f)$ to $\partial_\tau(f')$; **in**-oriented (resp. **out**-oriented) tails of τ are called *inputs* (resp. *outputs*) of τ ; if v is a vertex of τ , the *inputs* (resp. *outputs*) of v are the elements of the set $\mathbf{in}_\tau(v) = \partial_\tau^{-1}(v) \cap \mathbf{o}_\tau^{-1}(\mathbf{in})$ (resp. $\mathbf{out}_\tau(v) = \partial_\tau^{-1}(v) \cap \mathbf{o}_\tau^{-1}(\mathbf{out})$).
- An *ordered graph* $(\tau, <_\tau)$ is a graph τ together with an order $<_\tau$ on its flags.

Different structures on a graph τ can combine modularly, for instance τ can be endowed with a labeling ℓ_τ and an orientation \mathbf{o}_τ , so as to form a directed labeled graph $(\tau, \mathbf{o}_\tau, \ell_\tau)$; and conversely, a directed labeled graph $(\tau, \mathbf{o}_\tau, \ell_\tau)$ can be seen as a directed graph (τ, \mathbf{o}_τ) , forgetting the labeling ℓ_τ . Roughly, a graph is labeled (resp. colored) when labels are associated with its vertices (resp. edges and “hanging” edges). In a directed graph, an input (resp. output) of a vertex v is a—half or hanging—edge incoming in (resp. outgoing from) v .

Graphs can be depicted in diagrammatic form, vertices are represented by small rounded boxes and flags by wires touching their endpoint. As a graph is just a disjoint union of corollas glued by the involution, we only need to depict corollas (Figure 1, on the left) and place the two halves of an edge next to each other (Figure 1, on the right). In directed graphs, inputs of a corolla are depicted above the corolla, outputs below; arrows also show their orientation. The color of a flag f (if any) is written next to f . The label of a vertex v (if any) is written inside v . If ordered, flags of a corolla are depicted increasing from left to right. A dotted gray wire denote an arbitrary number (possibly 0) of flags (see Figures 8 and 9).

Example 3.4. The directed labeled colored ordered corolla $\mathbf{5} = (*_{\mathbf{5}}, \mathbf{o}_{\mathbf{5}}, \ell_{\mathbf{5}}, \mathbf{c}_{\mathbf{5}}, <_{\mathbf{5}})$ depicted in Figure 1 (on the left) has $*$ as its only vertex and $F_{\mathbf{5}} = \{0, 1, 2, 3, 4\}$ as its set of flags; it is endowed with the order $0 <_{\mathbf{5}} 4$ and $1 <_{\mathbf{5}} 2 <_{\mathbf{5}} 3$, the labeling $\ell_{\mathbf{5}}(*) = \mathbf{X}$, the orientation $\mathbf{o}_{\mathbf{5}}: F_{\mathbf{5}} \rightarrow \{\mathbf{in}, \mathbf{out}\}$ defined by $\mathbf{o}_{\mathbf{5}}(0) = \mathbf{o}_{\mathbf{5}}(4) = \mathbf{out}$ and $\mathbf{o}_{\mathbf{5}}(1) = \mathbf{o}_{\mathbf{5}}(2) = \mathbf{o}_{\mathbf{5}}(3) = \mathbf{in}$, the coloring $\mathbf{c}_{\mathbf{5}}: F_{\mathbf{5}} \rightarrow \{A_0, \dots, A_4\}$ defined by $\mathbf{c}(i) = A_i$ for all $i \in F_{\mathbf{5}}$.

Let $\sigma_{\mathbf{ax}}$ be the directed labeled colored corolla, whose only vertex is labeled by **ax**, and whose only flags are the outputs 5 (colored by A_2) and 6 (colored by A_3). The directed labeled colored ordered two-vertex graph ρ in Figure 1 (on the right) is obtained by “grafting” $\sigma_{\mathbf{ax}}$ into $\mathbf{5}$, *i.e.* from $\sigma_{\mathbf{ax}}$ and $\mathbf{5}$ by defining the involution $j_\rho: \{0, \dots, 6\} \rightarrow \{0, \dots, 6\}$ as $j_\rho(i) = j_{\mathbf{5}}(i)$ for $i \in \{0, 1, 4\}$, and $j_\rho(i) = i + 3$ for $i \in \{2, 3\}$, and $j_\rho(i) = i - 3$ for $i \in \{5, 6\}$.



FIGURE 1. A directed labeled colored ordered corolla $\mathbf{5}$ (left), and a directed labeled colored ordered two-vertex graph ρ (right), see Example 3.4.

Each enrichment of the graph structure introduced in Definition 3.3 induces a notion of structure-preserving morphism (and isomorphism) that extends the notion of graph morphism (and graph isomorphism) seen in Definition 3.2, so as to form an associated category. Moreover, different kinds of structure preservation can be combined in a modular way. For instance, given two directed labeled graphs $(\tau, \mathbf{o}_\tau, \ell_\tau)$ and $(\sigma, \mathbf{o}_\sigma, \ell_\sigma)$ both with labels in L , (in particular, they can be seen as two directed graphs (τ, \mathbf{o}_τ) and $(\sigma, \mathbf{o}_\sigma)$, forgetting labels), a *directed graph morphism* $h: (\tau, \mathbf{o}_\tau) \rightarrow (\sigma, \mathbf{o}_\sigma)$ is a graph morphism $h = (h_F, h_V): \tau \rightarrow \sigma$ such that $\mathbf{o}_\sigma \circ h_F = \mathbf{o}_\tau$; this means that h_F maps input (resp. output) flags of τ to input (resp. output) flags of σ . And a *directed labeled isomorphism* $h: (\tau, \mathbf{o}_\tau, \ell_\tau) \rightarrow (\sigma, \mathbf{o}_\sigma, \ell_\sigma)$ is a graph isomorphism $h = (h_F, h_V): \tau \rightarrow \sigma$ such that $\mathbf{o}_\sigma \circ h_F = \mathbf{o}_\tau$ and $\ell_\sigma \circ h_V = \ell_\tau$.

Trees and paths. An *undirected path* on a graph τ is a finite sequence of flags $\varphi = (f_1, \dots, f_{2n})$ for some $n \in \mathbb{N}$ such that, for all $1 \leq i \leq n$, $j_\tau(f_{2i-1}) = f_{2i} \neq f_{2i-1}$ and (if $i \neq n$) $\partial_\tau(f_{2i}) = \partial_\tau(f_{2i+1})$ with $f_{2i} \neq f_{2i+1}$. We say that φ is *between* $\partial_\tau(f_1)$ and $\partial_\tau(f_{2n})$ if $n > 0$ (and it is a *cycle* if moreover $\partial_\tau(f_1) = \partial_\tau(f_{2n})$), otherwise it is the *empty (undirected) path*, which is between any vertex and itself; the *length* of φ is n . Note that φ cannot cross any tail of τ . Two vertices in a graph are *connected* if there is an undirected path between them.

Let τ be a graph: τ is *connected* if any vertices $v, v' \in V_\tau$ are connected; a *connected component* of τ is a maximal (with respect to the inclusion of flags and vertices) connected subgraph of τ ; τ is *acyclic* (or a *forest*) if it has no cycles; τ is a *tree* if it is a connected forest. Note that a forest can be seen as a non-empty finite sequence of trees, each tree is a connected component; so, an empty forest can be seen as a non-empty finite sequence of empty trees.

A *rooted tree* τ is a non-empty directed tree such that each vertex has exactly one output. So, by finiteness, τ has exactly one output tail f : the endpoint of f is called the *root* of τ .

Remark 3.5. Let τ and τ' be two rooted trees, and $h: \tau \rightarrow \tau'$ be a directed graph morphism. As h_F preserves tails and orientation, h_V maps the root of τ to the root of τ' . Rooted trees and directed graph morphisms over them form a category **RoTree**.

A *directed path* on a directed graph τ is an undirected path $\varphi = (f_1, \dots, f_{2n})$ for some $n \in \mathbb{N}$ where f_{2i-1} is output and f_{2i} is input for all $1 \leq i \leq n$. We say that φ is *from* $\partial_\tau(f_1)$ to $\partial_\tau(f_{2n})$ if $n > 0$; otherwise it is the *empty (directed) path*, from any vertex to itself.

The set of directed paths on a directed tree τ is finite. As such, we define the *reflexive-transitive closure*, or *free category*, τ° of τ as the directed graph with same vertices and same tails as τ , and with an edge from v to v' for any directed path from v to v' in τ . The operator $(\cdot)^\circ$ lifts to a functor from the category **RoTree** to the category of directed graphs, so for every morphism f in **RoTree**, f° is a directed graph morphism between directed graphs.

Pullback in the category of graphs. The category of graphs has all pullbacks, a fact that we use extensively. We recall here all the definitions and facts involved in that affirmation.

Definition 3.6 (pullback). Let \mathcal{C} be a category. Let X, Y, Z be objects of \mathcal{C} , and $f: X \rightarrow Z$ and $g: Y \rightarrow Z$ be arrows of \mathcal{C} . A *pullback* of f and g is the triple $(P, !_X, !_Y)$ where P is an object of \mathcal{C} and $!_X: P \rightarrow X$ and $!_Y: P \rightarrow Y$ are arrows of \mathcal{C} such that diagram (3.1) commutes and, for any $(Q, h: Q \rightarrow X, k: Q \rightarrow Y)$ making the same diagram commute, there is a unique arrow $u: Q \rightarrow P$ factorizing h and k , *i.e.* such that diagram (3.2) commutes.

$$\begin{array}{ccc}
 P & \xrightarrow{!_X} & X \\
 !_Y \downarrow & & \downarrow f \\
 Y & \xrightarrow{g} & Z
 \end{array} \quad (3.1)
 \qquad
 \begin{array}{ccc}
 Q & \xrightarrow{h} & X \\
 \text{---} u \text{---} \searrow & & \downarrow f \\
 & P & \xrightarrow{!_X} & X \\
 & !_Y \downarrow & & \downarrow f \\
 & Y & \xrightarrow{g} & Z \\
 & \text{---} k \text{---} \searrow & &
 \end{array} \quad (3.2)
 \qquad
 \begin{array}{ccc}
 X \times_Z Y & \xrightarrow{!_X} & X \\
 !_Y \downarrow \lrcorner & & \downarrow f \\
 Y & \xrightarrow{g} & Z
 \end{array} \quad (3.3)$$

A pullback $(P, !_X, !_Y)$ of $f: X \rightarrow Z$ and $g: Y \rightarrow Z$ is unique (up to unique isomorphism); P is usually denoted by $X \times_Z Y$ (leaving f, g implicit) and depicted like in diagram (3.3).

All pullbacks exist in the category **Graph** of graphs. Let us show it explicitly. Let $\tau = (F_\tau, V_\tau, \partial_\tau, j_\tau)$, $\sigma = (F_\sigma, V_\sigma, \partial_\sigma, j_\sigma)$ and $\rho = (F_\rho, V_\rho, \partial_\rho, j_\rho)$ be graphs, and let $g = (g_F, g_V): \sigma \rightarrow \tau$ and $h = (h_F, h_V): \rho \rightarrow \tau$ be graph morphisms. Consider the sets

$$F = \{(f_\sigma, f_\rho) \in F_\sigma \times F_\rho \mid g_F(f_\sigma) = h_F(f_\rho)\} \quad V = \{(v_\sigma, v_\rho) \in V_\sigma \times V_\rho \mid g_V(v_\sigma) = h_V(v_\rho)\}.$$

They are both equipped with their natural projections $\pi_\sigma^F: F \rightarrow F_\sigma$, $\pi_\rho^F: F \rightarrow F_\rho$ and $\pi_\sigma^V: V \rightarrow V_\sigma$, $\pi_\rho^V: V \rightarrow V_\rho$. Let $f \in F$.

$$\begin{aligned}
 (g_V \circ \partial_\sigma \circ \pi_\sigma^F)(f) &= (\partial_\tau \circ g_F \circ \pi_\sigma^F)(f) && \text{because } g \text{ is a graph morphism} \\
 &= (\partial_\tau \circ h_F \circ \pi_\rho^F)(f) && \text{by definition of } F \\
 &= (h_V \circ \partial_\rho \circ \pi_\rho^F)(f) && \text{because } h \text{ is a graph morphism.}
 \end{aligned}$$

Hence, we can define $\partial: F \rightarrow V$ by $\partial(f) = ((\partial_\sigma \circ \pi_\sigma^F)(f), (\partial_\rho \circ \pi_\rho^F)(f))$. In the same way, we define $j: F \rightarrow F$ by $j(f) = ((j_\sigma \circ \pi_\sigma^F)(f), (j_\rho \circ \pi_\rho^F)(f))$, and check that it is an involution.

Thus, $\sigma \times_\tau \rho = (F, V, \partial, j)$ is a graph⁸ and $\pi_\sigma = (\pi_\sigma^F, \pi_\sigma^V): \sigma \times_\tau \rho \rightarrow \sigma$ and $\pi_\rho = (\pi_\rho^F, \pi_\rho^V): \sigma \times_\tau \rho \rightarrow \rho$ are graph morphisms such that diagram (3.4) below commutes.

$$\begin{array}{ccc}
 \sigma \times_\tau \rho & \xrightarrow{\pi_\sigma} & \sigma \\
 \pi_\rho \downarrow & & \downarrow g \\
 \rho & \xrightarrow{h} & \tau
 \end{array} \quad (3.4)
 \qquad
 \begin{array}{ccc}
 \mu & \xrightarrow{p} & \sigma \\
 q \downarrow & & \downarrow g \\
 \rho & \xrightarrow{h} & \tau
 \end{array} \quad (3.5)$$

Consider now any graph $\mu = (F_\mu, V_\mu, \partial_\mu, j_\mu)$ and graph morphisms $p = (p_F, p_V): \mu \rightarrow \sigma$ and $q = (q_F, q_V): \mu \rightarrow \rho$ such that diagram (3.5) above commutes. For every $f \in F_\mu$, let $r_F(f) = (p_F(f), q_F(f))$, and for every $v \in V_\mu$, let $r_V(v) = (p_V(v), q_V(v))$. It is easy to check that it defines a graph morphism $r = (r_F, r_V): \mu \rightarrow \sigma \times_\tau \rho$ that is the unique one to factorize p and q . Therefore, $(\sigma \times_\tau \rho, \pi_\sigma, \pi_\rho)$ is the pullback of g and h .

Roughly, the pullback $\sigma \times_\tau \rho$ is obtained as a sort of “lax intersection” of the “similar” vertices of σ and ρ (they are “similar” when they are sent to the same vertex in τ , by g_V and h_V respectively), and keeping whatever flags that are in the “intersection”. So, $\sigma \times_\tau \rho$ is the maximal graph that is “compatible” with both σ and ρ .

The pullback construction lifts to directed, labeled, colored and ordered graphs.

⁸Note that if σ or ρ is the empty graph, then so is $\sigma \times_\tau \rho$, and π_σ and π_ρ are empty graph morphisms.

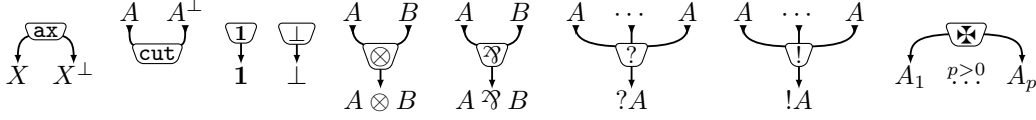


FIGURE 2. Cells, with their labels and their typed inputs and outputs.

4. MELL PROOF-STRUCTURES AND QUASI-PROOF-STRUCTURES

We present here a purely graphical definition of MELL proof-structures, following the non-inductive approach of [GPT19]: a MELL proof-structure R is essentially a labeled directed graph $|R|$ together with some additional information to identify its boxes. Our definition is completely based on standard notions (recalled in Section 3) from graph theory: it is formal (with an eye towards complete computer formalization) but avoids *ad hoc* technicalities to identify boxes. Indeed, the inductive and ordered structure of the boxes of R is recovered by means of a tree \mathcal{A}_R and a graph morphism box_R from $|R|$ to \mathcal{A}_R which allows us to recognize the content and the border of all boxes in R . We use n -ary vertices of type $?$ collapsing weakening, dereliction and contraction (like in [DR95]). In this way, we get a *canonical* representation of MELL proof-structures with respect to operations like associativity and commutativity of contractions, or neutrality of weakening with respect to contraction.

One of the benefits of our purely graphical definition of proof-structure is that notions involving proof-structures such as isomorphism (see below), Taylor expansion (see Section 5) and correctness graph (see [GPT19]) can be defined rigorously by means of standard (and not *ad hoc*) mathematical tools from graph theory, making them more “manageable”.

4.1. MELL formulas and lists. Given a countably infinite set of propositional variables X, Y, Z, \dots , MELL *formulas* are defined by the following inductive grammar:

$$A, B ::= X \mid X^\perp \mid \mathbf{1} \mid \perp \mid A \otimes B \mid A \wp B \mid !A \mid ?A \quad (\text{set: } \mathcal{Form}_{\text{MELL}})$$

Linear negation $(\cdot)^\perp$ is defined via De Morgan laws $\mathbf{1}^\perp = \perp$, $(A \otimes B)^\perp = A^\perp \wp B^\perp$ and $(!A)^\perp = ?A^\perp$, so as to be involutive, *i.e.* $A^{\perp\perp} = A$ for any MELL formula A (we say that A and A^\perp are *dual*). Variables and their negations are *atomic* formulas; \otimes and \wp (resp. $!$ and $?$) are *multiplicative* (resp. *exponential*) *connectives*; $\mathbf{1}$ and \perp are (*multiplicative*) *units*.

We call any finite sequence a *list*. The *empty list* is denoted by ϵ . A list (x_1, \dots, x_n) with $n > 0$ and $x_i = \epsilon$ for all $1 \leq i \leq n$ is denoted by $n\epsilon$ or $\vec{\epsilon}$. Note that a list $\vec{\epsilon}$ is not empty.

Let $m, n \in \mathbb{N}$ and $\{i_j\}_{0 \leq j \leq n} \subseteq \mathbb{N}$ with $i_j < i_{j+1}$, $i_0 = 0$, $i_n = m$. Let A_1, \dots, A_m be MELL formulas, let $\Gamma_j = (A_{i_{j-1}+1}, \dots, A_{i_j})$ for all $1 \leq j \leq n$. The list $\Gamma = (\Gamma_1, \dots, \Gamma_n)$ of lists of MELL formulas is also denoted by $(A_1, \dots, A_{i_1}; \dots; A_{i_{n-1}+1}, \dots, A_m)$, with lists separated by semicolons. The *flattening* of Γ is the list (A_1, \dots, A_m) of MELL formulas.

4.2. Proof-structures. We define here proof-structures corresponding to some subsystems and extensions of LL: MELL, DiLL and DiLL₀. Full differential linear logic (DiLL) is an extension of MELL (with the same language as MELL) provided with both promotion rule (*i.e.*, boxes) and co-structural rules for the $!$ -modality (the duals of the structural rules handling the $?$ -modality): DiLL₀ and MELL are particular subsystems of DiLL, respectively the promotion-free one (*i.e.*, without boxes) and the one without co-structural rules. We reuse the syntax of proof-structures given in [GPT19], based on the graph notions introduced

in Section 3. Note that, unlike [PT09], we allow the presence of cuts (vertices of type **cut**), even though we are not interested in cut-elimination and its study is left to future work.

Definition 4.1 (module, proof-structure). A (DiLL) *module* $M = (|M|, \ell, \mathbf{o}, \mathbf{c}, <)$ is a labeled (ℓ), directed (\mathbf{o}), colored (\mathbf{c}), ordered ($<$) graph $|M|$ without loops such that:

- the map $\ell: V_{|M|} \rightarrow \{\mathbf{ax}, \mathbf{cut}, \mathbf{1}, \perp, \otimes, \wp, ?, !, \boxtimes\}$ associates with each vertex v its *type* $\ell(v)$;
- the map $\mathbf{c}: F_{|M|} \rightarrow \mathcal{F}orm_{\text{MELL}}$ associates with each flag f its *type* $\mathbf{c}(f)$;
- $<$ is a strict partial order on the flags of $|M|$ that is total on the tails of $|M|$ and on the inputs of each vertex of type \wp or \otimes , separately;
- for every vertex $v \in V_{|M|}$,
 - if $\ell(v) = \mathbf{cut}$, v has no output and two inputs i_1 and i_2 , such that $\mathbf{c}(i_1) = \mathbf{c}(i_2)^\perp$;
 - if $\ell(v) = \mathbf{ax}$, v has no inputs and two outputs o_1 and o_2 , with $\mathbf{c}(o_1) = \mathbf{c}(o_2)^\perp$ atomic;⁹
 - if $\ell(v) \in \{\mathbf{1}, \perp\}$, v has no inputs and only one output o , with $\mathbf{c}(o) = \ell(v)$;
 - if $\ell(v) \in \{\otimes, \wp\}$, v has two inputs $i_1 < i_2$ and one output o , with $\mathbf{c}(o) = \mathbf{c}(i_1) \ell(v) \mathbf{c}(i_2)$;
 - if $\ell(v) \in \{?, !\}$, v has $n \geq 0$ inputs i_1, \dots, i_n and one output o , such that $\mathbf{c}(o) = \ell(v) \mathbf{c}(i_j)$ for all $1 \leq j \leq n$;¹⁰ v is a *contraction* (resp. *co-contraction*) if $\ell(v) = ?$ (resp. $\ell(v) = !$);
 - if $\ell(v) = \boxtimes$, v has no inputs and $p > 0$ outputs o_1, \dots, o_p .¹¹

In Figure 2 we depicted the corollas associated with all types of vertices.

A (DiLL) *proof-structure* is a triple $R = (|R|, \mathcal{A}, \mathbf{box})$ where:

- $|R| = (\|R\|, \ell_R, \mathbf{o}_R, \mathbf{c}_R, <_R)$ is a module with no input tails, called the *structured graph* of R , and $\|R\| = (F_{\|R\|}, V_{\|R\|}, \partial_{\|R\|}, j_{\|R\|})$ is the (*undirected*) *graph* of R ;
- \mathcal{A} is a rooted tree with no input tails, called the *box-tree* of R ;¹²
- $\mathbf{box} = (\mathbf{box}_F, \mathbf{box}_V): |R| \rightarrow \mathcal{A}^\circ$ is a directed graph morphism,¹³ called the *box-function* of R , such that \mathbf{box}_F induces a bijection from a subset of $\bigcup_{v \in V_{\|R\|}, \ell(v) = !} \mathbf{in}_{|R|}(v)$ (the set of inputs of the vertices of type $!$ in $|R|$) to the set of input flags in \mathcal{A} ;¹⁴ and for any $v \in V_{\|R\|}$ and any input f of v , if $\mathbf{box}_V((\partial_{\|R\|} \circ j_{\|R\|})(f)) \neq \mathbf{box}_V(\partial_{\|R\|}(f))$ then $\ell(v) \in \{!, ?\}$.¹⁵

A (DiLL) proof-structure $R = (|R|, \mathcal{A}, \mathbf{box})$ is said to be:

- (1) MELL if its structured graph $|R|$ has no vertices of type \boxtimes and
 - all vertices in $|R|$ of type $!$ have exactly one input,

⁹We deal with *atomic axioms* only to simplify the presentation in the next sections. The general case with non-atomic axioms (*i.e.* $\mathbf{c}(o_1)$ and $\mathbf{c}(o_2)$ are still dual but not necessarily atomic) is discussed in Section 10.

¹⁰This implies that $\mathbf{c}(i_j) = \mathbf{c}(i_k)$ for all $1 \leq j, k \leq n$. There are no conditions on the number n of inputs.

¹¹There are no conditions on the number of outputs o_1, \dots, o_p (besides $p > 0$) or on their types.

¹²Intuitively, \mathcal{A} represents the tree-structure of the nested boxes of R , see Remark 4.2 below.

¹³The structured graph $|R|$ of R is more structured than a directed graph such as \mathcal{A}° : $|R|$ is directed but also labeled, colored, ordered. When we talk of a morphism between two structured graphs where one of the two, say σ , is less structured than the other, say τ , we mean that τ must be only considered with the same structure as σ . Thus, in this case, \mathbf{box} is a morphism from $(\|R\|, \mathbf{o}_R)$ —discarding $\ell_R, \mathbf{c}_R, <_R$ —to \mathcal{A}° .

¹⁴It means that for any input flag f' in \mathcal{A} there is exactly one input f of some vertex of type $!$ in $|R|$ such that $\mathbf{box}_F(f) = f'$; but, for any input f of some vertex of type $!$ in $|R|$, $\mathbf{box}_F(f)$ need not be an input flag in \mathcal{A} (by definition of directed graph morphism, $\mathbf{box}_F(f)$ is necessarily an input flag in \mathcal{A}°). In particular, if $|R|$ has no vertices of type $!$, \mathcal{A} is a root with its output and no inputs; and if $\|R\|$ is the empty graph, \mathbf{box} is the empty graph morphism from $|R|$ to such a \mathcal{A} . Intuitively, given a vertex v of type $!$ in $|R|$, a box is associated with (and only with) each input f of v such that $\mathbf{box}_F(f)$ is an input flag in \mathcal{A} and not only in \mathcal{A}° : f represents the principal door in the border of such a box, indeed by definition of directed graph morphism, $\mathbf{box}_V((\partial_{\|R\|} \circ j_{\|R\|})(f)) \neq \mathbf{box}_V(\partial_{\|R\|}(f))$, and for all $f' \in F_{\|R\|}$, if $f' \neq f$ then $\mathbf{box}_F(f') \neq \mathbf{box}_F(f)$.

¹⁵Roughly, it says that the border of a box is made of inputs of vertices of type $!$ or $?$ in $|R|$.

- the bijection induced by \mathbf{box}_F is defined on the whole set $\bigcup_{v \in V_{\|R\|}, \ell(v)=!} \mathbf{in}_{|R|}(v)$;¹⁶
- (2) \mathbf{DiLL}_0 (or *resource*) if \mathcal{A} consists only of the root with its output, and either $|R|$ has no vertices of type \boxtimes or $|R|$ is a *daimon*, i.e. $|R|$ has only one vertex and it is of type \boxtimes ;
- (3) *empty* (which is both \mathbf{MELL} and \mathbf{DiLL}_0) if its graph $\|R\|$ is empty; it is denoted by ε ;
- (4) *cut-free* if its structured graph $|R|$ has no vertices of type *cut*, otherwise it is *with cuts*.

Our \mathbf{MELL} proof-structures correspond to usual \mathbf{MELL} proof-structures (as in [dCT12]). Our \mathbf{DiLL}_0 proof-structures correspond to usual \mathbf{DiLL}_0 proof-structures (as in [ER06b]) except that we allow also (proof-structures that are) *daimons*. Daimons will be used in the Taylor expansion to deal with the content of a box taken 0 times (see Section 5). Our interest for \mathbf{DiLL} proof-structures is just to have a unitary syntax subsuming both \mathbf{MELL} and \mathbf{DiLL}_0 without considering cut-elimination: for this reason, unlike [Pag09, Tra09, Tra11], our \mathbf{DiLL} proof-structures are not allowed to contain a sum of \mathbf{DiLL} proof-structure inside a box.¹⁷

Given a proof-structure $R = (|R|, \mathcal{A}, \mathbf{box})$, the output tails of $|R|$ are the *conclusions* of R . The *type* of R is the list of the types of these conclusions, ordered according to $<_{|R|}$.

Borrowing the terminology from [ER06b, Laf90], in proof-structures, we speak of *cells* instead of vertices, and of ℓ -*cell* for a cell of type ℓ . An *hypothesis* is a cell without inputs.

Remark 4.2 (box). In our syntax, boxes do not have explicit constructors or cells, hence boxes and depth of a proof structure $R = (|R|, \mathcal{A}, \mathbf{box})$ are recovered in a non-inductive way.

Roughly, any non-root vertex v in \mathcal{A} induces a subgraph of \mathcal{A}° made up of all vertices “above v ” with their inputs and outputs: the preimage of this subgraph through \mathbf{box} is the *box* of v in $|R|$. The preimage of the root of \mathcal{A} through \mathbf{box} is the part of $|R|$ outside any box.

More precisely, with every flag f of $|R|$ such that $\mathbf{box}_F(f)$ is an input flag of \mathcal{A} ¹⁸ is associated a *box* B_f , that is the subgraph of $|R|$ (which is actually a proof-structure) made up of all the cells v (with their inputs and outputs) such that there is a directed path on \mathcal{A} from $\mathbf{box}_V(v)$ to $\mathbf{box}_V((\partial_{\|R\|} \circ j_{\|R\|})(f))$ (note that f and the $!$ -cell of which f is an input are not in B_f). A *conclusion* of such a box B_f is any output flag f' in B_f such that $(\partial_{\|R\|} \circ j_{\|R\|})(f')$ is not in B_f . The tree-structure of \mathcal{A} expresses the usual *nesting condition* of boxes: two boxes in $|R|$ are either disjoint or contained one in the other.

The *depth* of a cell v of R is the length of the directed path in \mathcal{A} from $\mathbf{box}_V(v)$ to the root of \mathcal{A} . The *depth* of R is the maximal depth of the cells of R .

Example 4.3. In Figure 3 a \mathbf{MELL} proof-structure $S = (|S|, \mathcal{A}_S, \mathbf{box}_S)$ is depicted. The box-function \mathbf{box}_S is kept implicit by means of colors: the colored areas in $|S|$ represent boxes (the preimages of non-root vertices of \mathcal{A}_S through \mathbf{box}_S), and the same color is used on \mathcal{A}_S to show where each box is mapped by \mathbf{box}_S .

¹⁶It means that for any input flag f' in \mathcal{A} there is exactly one vertex of type $!$ in $|R|$ whose unique input f is such that $\mathbf{box}_F(f) = f'$; and, if f is the (only) input some vertex of type $!$ in $|R|$, $\mathbf{box}_F(f)$ is an input flag in \mathcal{A} . Intuitively, a box is associated with (and only with) the unique input of each vertex of type $!$ in $|R|$.

¹⁷This restriction is without loss of generality, because a box containing a sum of proof-structures is equivalent (see [Tra09]) to a co-contraction of the boxes of each proof-structure in the sum. Also, the vertex of type \boxtimes corresponds to the empty sum inside a box: roughly, it is the content of an “empty box”.

¹⁸By the constraints on \mathbf{box} , this condition can be fulfilled only by inputs of $!$ -cells in $|R|$, and an input of a $!$ -cell need not fulfill it; in particular, if R is a \mathbf{MELL} proof-structure, then this condition is fulfilled by all and only the inputs of $!$ -cells (and such an input is unique for any $!$ -cell) in $|R|$; but if R is a \mathbf{DiLL}_0 proof-structure, this condition is not fulfilled by any flag in $|R|$ (since \mathcal{A} has no inputs) and so \mathbf{box} is a directed graph morphism associating the root of \mathcal{A} with any cell of R . Thus, in a \mathbf{DiLL}_0 proof-structure $\rho = (|\rho|, \mathcal{A}_\rho, \mathbf{box}_\rho)$, there are no boxes, \mathcal{A}_ρ and \mathbf{box}_ρ do not induce any structure on $|\rho|$: ρ can be identified with $|\rho|$.

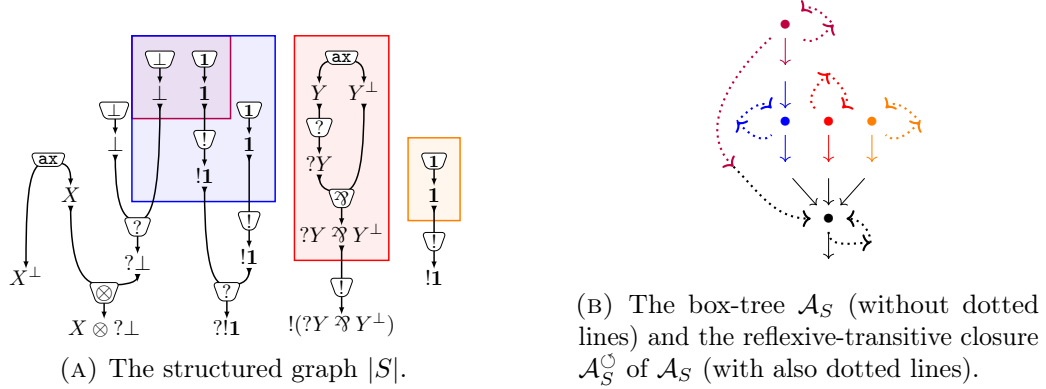


FIGURE 3. A MELL proof-structure $S = (|S|, \mathcal{A}_S, \text{box}_S)$.

The proof-structures we have defined depend on the carrier sets for vertices and flags. It is natural to abstract away from them, and identify two proof-structures that differ only in the name of their vertices and flags, through a handy notion of isomorphism (*i.e.* structure-preserving bijection) inherited from the one of graph isomorphism defined in Section 3.

Definition 4.4 (isomorphism). Let $R = (|R|, \mathcal{A}, \text{box})$ and $R' = (|R'|, \mathcal{A}', \text{box}')$ be proof-structures. An *isomorphism* from R to R' is a couple $(\varphi_{|\cdot|}, \varphi_{\text{tree}})$ where $\varphi_{|\cdot|}: |R| \rightarrow |R'|$ is a structured graph isomorphism, and $\varphi_{\text{tree}}: \mathcal{A} \rightarrow \mathcal{A}'$ is a directed graph isomorphism, such that the diagram below commutes.

$$\begin{array}{ccc} |R| & \xrightarrow{\text{box}} & \mathcal{A}^\circ \\ \downarrow \varphi_{|\cdot|} & & \downarrow \varphi_{\text{tree}}^\circ \\ |R'| & \xrightarrow{\text{box}'} & \mathcal{A}'^\circ \end{array}$$

Note that if R is a MELL or DiLL₀ proof-structure isomorphic to a proof-structure R' , then R' is respectively MELL or DiLL₀. Often we implicitly consider proof-structures up to isomorphism (as we do, for instance, in every figure representing proof-structures).

4.3. Quasi-proof-structures. We need to consider more general structures in order to accommodate our rewrite rules, as discussed in Section 2, p. 5 (the rewrite rules are defined in Sections 6 and 7). This is why we extend all the definitions to *tuples* of proof-structures.

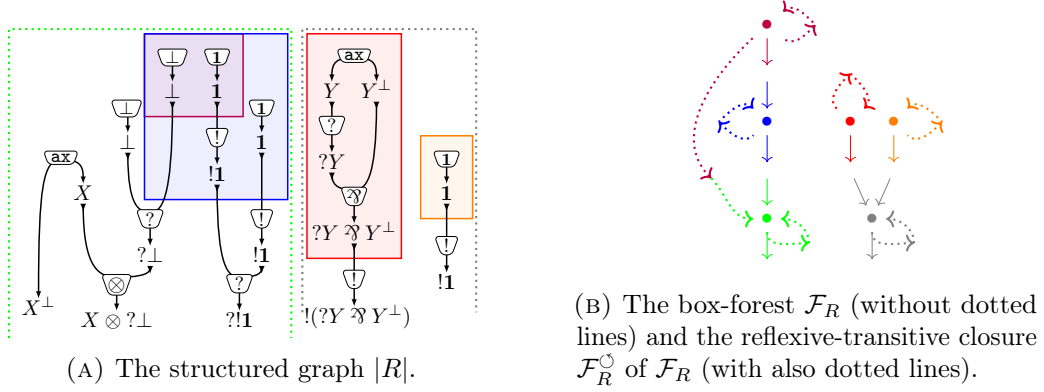
Definition 4.5 (quasi-proof-structure). A (DiLL) *quasi-proof-structure* is a tuple $R = (R_1, \dots, R_n)$ of proof-structures, for some $n > 0$; for all $1 \leq i \leq n$, R_i is a *component* of R . If $R_i = \varepsilon$ for all $1 \leq i \leq n$, then R is an *empty quasi-proof-structure*, noted $n\varepsilon$ or $\bar{\varepsilon}$.¹⁹

For convenience, given a proof-structure $R_i = (|R_i|, \mathcal{A}_i, \text{box}_i)$ for all $1 \leq i \leq n$, we denote the quasi-proof-structure $R = (R_1, \dots, R_n)$ as $R = (|R|, \mathcal{F}_R, \text{box}_R)$, where $|R| = (|R_1|, \dots, |R_n|)$ is the *structured graph* of R , $\mathcal{F}_R = (\mathcal{A}_1, \dots, \mathcal{A}_n)$ is the *box-forest* of R , $\text{box}_R = (\text{box}_1, \dots, \text{box}_n)$ is the *box-function* of R ; the *graph* of R is $\|R\| = (\|R_1\|, \dots, \|R_n\|)$.

A *conclusion* of R is any conclusion of any component of R .

A quasi-proof-structure R is MELL (resp. DiLL₀) if all components of R are MELL (resp. DiLL₀) proof-structures.

¹⁹The 0-ary tuple of proof-structures is *not* a quasi-proof-structure. Indeed, every quasi-proof-structure has at least one (possibly empty) component. Note that the components of a quasi-proof-structure are independent of each other, in particular they may differ in the number of their conclusions or in their type.


 FIGURE 4. A MELL quasi-proof-structure $R = (|R|, \mathcal{F}_R, \text{box}_R)$.

The short notation $R = (|R|, \mathcal{F}_R, \text{box}_R)$ for quasi-proof-structures makes sense because the structured graph $|R| = (|R_1|, \dots, |R_n|)$ can be seen as the disjoint union of its components $|R_1|, \dots, |R_n|$, and similarly for \mathcal{F}_R and box_R . In particular, the box-forest \mathcal{F}_R is the disjoint union of the box-trees $\mathcal{A}_1, \dots, \mathcal{A}_n$ of R_1, \dots, R_n . The box-function box_R locates not only the boxes on $|R|$, but also the different components of R on $|R|$ (see also Remark 4.7).

The only delicate point is the definition of the order $<_{|R|}$ for the conclusions of the quasi-proof-structure $R = (R_1, \dots, R_n)$, given the order $<_{|R_i|}$ for each component $|R_i|$. Given the conclusions f, f' of R , we set $f <_{|R|} f'$ if either f is a conclusion of R_i and f' is a conclusion of $R_{i'}$ with $i < i'$, or f and f' are conclusions of the same component R_j and $f <_{|R_j|} f'$.

We often identify the conclusions of R and their order $<_{|R|}$ with a finite initial segment of $\mathbb{N} \setminus \{0\}$ and its natural order. Note that $\vec{\varepsilon}$ has no conclusions, akin to ε , as $\|\varepsilon\|$ is empty.

The *type* of a quasi-proof-structure $R = (R_1, \dots, R_n)$ is a list $\Gamma = (\Gamma_1, \dots, \Gamma_n)$ of lists of MELL formulas such that Γ_i is the type of R_i for all $1 \leq i \leq n$. When $n = 1$, we often identify R and R_1 (a quasi-proof-structure with only one component and a proof-structure), or Γ and Γ_1 (the type of a quasi-proof-structure with only one component and the type of a proof-structure). So, the type of a quasi-proof-structure R determines if R is a proof-structure.

The type of an empty quasi-proof-structure $\vec{\varepsilon}$ is a non-empty list of empty lists of MELL formulas, where each occurrence of the empty list of MELL formulas is the type of a component ε of $\vec{\varepsilon}$. More precisely, ε is the type of ε , and $n\varepsilon$ is the type of $n\varepsilon$ for every $n > 0$.

Example 4.6. In Figure 4 a MELL quasi-proof-structure R is depicted. The colored areas represent the preimages of boxes, and each dashed box represents a component of R (in other examples, the absence of a dashed line means that there is only one component).

Remark 4.7 (components). A difference arises between a DiLL_0 proof-structure and a DiLL_0 quasi-proof-structure. Both have no boxes. In the former, its box-tree and box-function do not induce any structure on its structured graph, so we can identify a DiLL_0 proof-structure with its structured graph. In the latter, its box-forest and box-function do induce a structure on its structured graph: indeed, its box-forest is made up only of roots with their output and its box-function separates the components of its structured graph, by mapping the vertices to the roots. So, we cannot identify a DiLL_0 quasi-proof-structure with its structured graph.

5. THE TAYLOR EXPANSION

The *Taylor expansion* $\mathcal{T}(R)$ [ER08] of a MELL (or more generally DiLL) proof-structure R is a possibly infinite set of DiLL₀ proof-structures: roughly, each element of $\mathcal{T}(R)$ is obtained from R by approximating—*i.e.* replacing—each box B in R with n_B copies of its content (for some $n_B \in \mathbb{N}$), recursively on the depth of R . Note that n_B depends not only on B but also on which “copy” of all boxes containing B we are considering. Usually, the Taylor expansion $\mathcal{T}(R)$ of a MELL proof-structure R is defined globally and inductively [MP07, PT09, CV18]: with R is directly associated the whole set $\mathcal{T}(R)$ by induction on the depth of R . A drawback of this approach is that, for each element of $\mathcal{T}(R)$, the way the copies of the content of a box are merged is defined “by hand”, which is syntactically heavy.

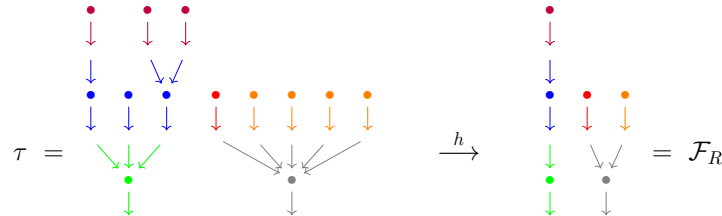
Following [GPT19], we adopt an alternative non-inductive approach, which strongly refines [GPT16]: the Taylor expansion is defined *pointwise* (see Example 5.2 and Figure 5). Indeed, proof-structures have a tree structure made explicit by their box-function. The definition of the Taylor expansion uses this tree structure: first, we define how to “*expand*” a tree via the notion of thick subtree [Bou09] (Definition 5.1; roughly, it states the number of copies of each box to be taken, recursively), we then take all these expansions of the box-tree of a proof-structure and we *pull* them *back* to the underlying graphs (Definition 5.4), finally we *forget* the tree structures associated with them (Definition 5.5). Thus, pullbacks give an abstract and elegant way to define the merging of copies of the content of a box in an element of $\mathcal{T}(R)$. The use of pullbacks is made possible because all the ingredients in our definition of a proof-structure live in the category of directed graphs (as defined in Section 3).

An advantage of our approach is that it smoothly generalizes to quasi-proof-structures.

Definition 5.1 (thick subtree and subforest). Let σ be a rooted tree. A *thick subtree* of σ is a pair (τ, h) of a rooted tree τ and a directed graph morphism $h = (h_F, h_V): \tau \rightarrow \sigma$.

Let $\sigma = (\sigma_1, \dots, \sigma_n)$ be a forest of rooted trees, with $n > 0$. A *thick subforest* of σ is a tuple $((\tau_1, h_1), \dots, (\tau_n, h_n))$ where (τ_i, h_i) is a thick subtree of σ_i for all $1 \leq i \leq n$. It is denoted by (τ, h) , where $\tau = (\tau_1, \dots, \tau_n)$ and $h = (h_1, \dots, h_n)$.

Example 5.2. The following is a graphical presentation of a thick subforest (τ, h) of the box-forest \mathcal{F}_R of the quasi-proof-structure in Figure 4, where the directed graph morphism $h = (h_F, h_V): \tau \rightarrow \mathcal{F}_R$ is depicted chromatically (same color means same image via h).



Intuitively, it means that τ is obtained from \mathcal{F}_R by taking 3 copies of the blue box, 1 copy of the red box and 4 copies of the orange box; in the first (resp. second; third) copy of the blue box, 1 copy (resp. 0 copies; 2 copies) of the purple box has been taken.

Remark 5.3 (roots). If (τ, h) is a thick subforest of a forest σ of rooted trees, h sets up a bijection between the roots of τ and σ , by definition of directed graph morphism; so, we can identify the roots of τ with the ones of σ . In particular, if σ is made of roots with their output and no inputs, the only thick subforest of σ is σ with its identity graph morphism.

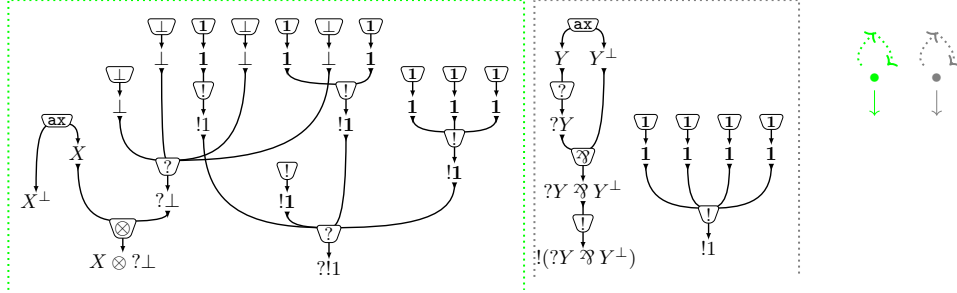


FIGURE 5. An element ρ of the Taylor expansion of the MELL quasi-proof-structure R in Figure 4, obtained from the element of $\mathcal{T}^{\text{Pr}}(R)$ in Example 5.2.

The crucial point is to pull back the expansion of a forest to quasi-proof-structures.

Definition 5.4 (proto-Taylor expansion). Let $R = (|R|, \mathcal{F}_R, \text{box}_R)$ be a quasi-proof-structure. The *proto-Taylor expansion* of R is the set $\mathcal{T}^{\text{Pr}}(R)$ of thick subforests of \mathcal{F}_R .

Let $t = (\tau_t, h_t) \in \mathcal{T}^{\text{Pr}}(R)$. The t -*expansion* of R is the pullback (R_t, p_t, p_R) below, computed in the category of directed graphs and directed graph morphisms.²⁰

$$\begin{array}{ccc} R_t & \xrightarrow{p_t} & \tau_t^\circ \\ p_R \downarrow & \lrcorner & \downarrow h_t^\circ \\ |R| & \xrightarrow{\text{box}_R} & \mathcal{F}_R^\circ \end{array}$$

Given a quasi-proof-structure R and $t = (\tau_t, h_t) \in \mathcal{T}^{\text{Pr}}(R)$, the directed graph R_t inherits the types of its vertices and flags by pre-composition of $\ell_{|R|}$ and $\text{c}_{|R|}$ with the graph morphism $p_R: R_t \rightarrow |R|$. The order on the flags of R_t is induced by the one on $|R|$ via p_R .

Let $[\tau_t]$ be the forest made up of the roots of τ_t and $\iota: \tau_t \rightarrow [\tau_t]$ be the graph morphism sending each vertex of τ_t to the root below it; ι° induces by post-composition a morphism $\bar{h}_t = \iota^\circ \circ p_t: R_t \rightarrow [\tau_t]^\circ$. The triple $(R_t, [\tau_t], \bar{h}_t)$ is a DiLL_0 quasi-proof-structure, and it is a DiLL_0 proof-structure if R is a proof-structure. We can now define the *Taylor expansion* $\mathcal{T}(R)$ of a quasi-proof-structure R (an example of an element of a Taylor expansion is in Figure 5).

Definition 5.5 (Taylor expansion). Let R be a quasi-proof-structure. The *Taylor expansion* of R is the set of DiLL_0 quasi-proof-structures $\mathcal{T}(R) = \{(R_t, [\tau_t], \bar{h}_t) \mid t = (\tau_t, h_t) \in \mathcal{T}^{\text{Pr}}(R)\}$.

An element $(R_t, [\tau_t], \bar{h}_t)$ of the Taylor expansion of a quasi-proof-structure R has much less structure than the pullback (R_t, p_t, p_R) : the latter indeed is a DiLL_0 quasi-proof-structure R_t coming with its projections $|R| \xleftarrow{p_R} R_t \xrightarrow{p_t} \tau_t^\circ$, which establish a precise correspondence between cells and flags of R_t and cells and flags of R : a cell in R_t is labeled (via the projections) by both the cell of $|R|$ and the branch of the box-forest of R it arose from. But $(R_t, [\tau_t], \bar{h}_t)$ where R_t is without its projections p_t and p_R loses the correspondence with R .

Remark 5.6 (conclusions). By definition and Remark 5.3, the Taylor expansion preserves conclusions and type: each element of the Taylor expansion of a quasi-proof-structure R has the *same conclusions* and the *same type* as R . More precisely, there is a bijection φ from the conclusions of a quasi-proof-structure R to the ones in each element ρ of $\mathcal{T}(R)$ such that

²⁰So, $|R|$ is considered as directed graph (see Footnote 13), forgetting that it is colored, labeled, ordered.

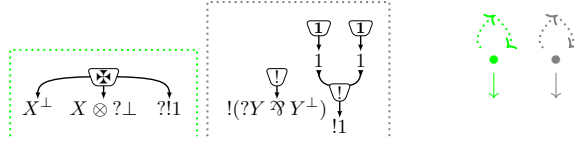


FIGURE 6. An element of the filled Taylor expansion of the MELL quasi-proof-structure R in Figure 4, obtained as an emptying of ρ in Figure 5.

i and $\varphi(i)$ have the same type and the same root (i.e. $\text{box}_{R_F}(i) = \text{box}_{\rho_F}(\varphi(i))$ up to the bijection of Remark 5.3). So, the types of R and ρ are the same (as a list of lists).

Example 5.7 (Taylor of the empty). By Definitions 4.1 and 4.5, the box-tree of ε is just a root with its output and no inputs (we call it a *root-tree*), and the box-forest of $n\varepsilon$ is a list of n root-trees, for all $n > 0$; the box-function of ε is the empty graph morphism and the box-function of $n\varepsilon$ is a list of n such morphisms. By Remark 5.3, $\mathcal{T}^{\text{Pr}}(\varepsilon)$ (resp. $\mathcal{T}^{\text{Pr}}(n\varepsilon)$ for all $n > 0$) is $\{(\tau, h)\}$ where τ is a root-tree (resp. a forest of n root-trees) and h is the identity graph morphism on τ . By Definition 5.5, $\mathcal{T}(\varepsilon) = \{\varepsilon\}$ and $\mathcal{T}(n\varepsilon) = \{n\varepsilon\}$ for all $n > 0$.

5.1. The filled Taylor expansion. As discussed in Section 2 (p. 6), the rewriting rules we will introduce in Section 6 need to “represent” the emptiness introduced by the Taylor expansion (taking 0 copies of a box) so as to preserve the conclusions. Thus, an element of the *filled Taylor expansion* $\mathcal{T}^{\text{F}}(R)$ of a quasi-proof-structure R (an example is in Figure 6) is obtained from an element of $\mathcal{T}(R)$ where some components can be erased and replaced by daimons with the same conclusions (hence $\mathcal{T}(R) \subseteq \mathcal{T}^{\text{F}}(R)$). The filled Taylor expansion (and not the plain one) will play a crucial role in Section 7 to define a natural transformation.

Definition 5.8 (emptying, filled Taylor expansion). The *emptying* of a DiLL_0 proof-structure ρ with at least one conclusion is a daimon whose conclusions and type are the same as ρ .

An *emptying* of a DiLL_0 quasi-proof-structure ρ is a DiLL_0 quasi-proof-structure obtained from ρ by replacing some (possibly none) components of ρ having at least one conclusion with their emptying.

The *filled Taylor expansion* $\mathcal{T}^{\text{F}}(R)$ of a quasi-proof-structure R is the set of all the emptyings of all the elements of its Taylor expansion $\mathcal{T}(R)$.

If $\rho = (R_t, [\tau_t], \bar{h}_t) \in \mathcal{T}(R)$, and r_1, \dots, r_n are some roots of $[\tau_t]$, the *emptying of ρ on r_1, \dots, r_n* , denoted by $\rho_{r_1 \dots r_n}$, is the element of $\mathcal{T}^{\text{F}}(R)$ obtained by substituting a daimon for each component of ρ with at least one conclusion corresponding to the root r_i (i.e., if such a conclusion is an output of a vertex v , then $\bar{h}_{tV}(v) = r_i$), for all $1 \leq i \leq n$.

Remark 5.9 (conclusions of emptying). By construction, the filled Taylor expansion preserves conclusions and type, as the Taylor expansion does (Remark 5.6): conclusions and type of a quasi-proof-structure R and of *any* emptying of *any* element of $\mathcal{T}(R)$ are the same.

Example 5.10 (filled Taylor of the empty). From Example 5.7 and Definition 5.8, it follows that $\mathcal{T}(\varepsilon) = \{\varepsilon\} = \mathcal{T}^{\text{F}}(\varepsilon)$ and for every $n > 0$, $\mathcal{T}(n\varepsilon) = \{n\varepsilon\} = \mathcal{T}^{\text{F}}(n\varepsilon)$.

Remark 5.11 (connection). The Taylor expansion does not “create connection”; the filled Taylor expansion does, but only in a component that has been filled by a daimon. More precisely, let i and j be conclusions of a quasi-proof-structure $R = (|R|, \mathcal{F}_R, \text{box}_R)$, let $\rho = (|\rho|, \mathcal{F}_\rho, \text{box}_\rho) \in \mathcal{T}(R)$ and let $\rho_{r_1 \dots r_n} \in \mathcal{T}^{\text{F}}(R)$ be the emptying of ρ on the roots r_1, \dots, r_n . If

$$\begin{array}{l}
(\Gamma_1; \dots; \Gamma_k, \mathbf{c}(i), \mathbf{c}(i+1), \Gamma'_k; \dots; \Gamma_n) \xrightarrow{\text{exc}_i} (\Gamma_1; \dots; \Gamma_k, \mathbf{c}(i+1), \mathbf{c}(i), \Gamma'_k; \dots; \Gamma_n) \\
(\Gamma_1; \dots; \Gamma_k, \mathbf{c}(i), \mathbf{c}(i+1), \Gamma'_k; \dots; \Gamma_n) \xrightarrow{\text{mix}_i} (\Gamma_1; \dots; \Gamma_k, \mathbf{c}(i); \mathbf{c}(i+1), \Gamma'_k; \dots; \Gamma_n) \\
(\Gamma_1; \dots; \Gamma_{k-1}; \mathbf{c}(i-1), \mathbf{c}(i); \Gamma_{k+1}; \dots; \Gamma_n) \xrightarrow{\text{ax}_i} (\Gamma_1; \dots; \Gamma_{k-1}; \epsilon; \Gamma_{k+1}; \dots; \Gamma_n) \text{ if } \mathbf{c}(i-1) = \mathbf{c}(i)^\perp \text{ atomic} \\
(\Gamma_1; \dots; \Gamma_k; \dots; \Gamma_n) \xrightarrow{\text{cut}^i} (\Gamma_1; \dots; \Gamma_k, \mathbf{c}(i), \mathbf{c}(i+1); \dots; \Gamma_n) \text{ if } \mathbf{c}(i) = \mathbf{c}(i+1)^\perp \\
(\Gamma_1; \dots; \Gamma_{k-1}; \mathbf{c}(i); \Gamma_{k+1}; \dots; \Gamma_n) \xrightarrow{\mathbf{1}_i} (\Gamma_1; \dots; \Gamma_{k-1}; \epsilon; \Gamma_{k+1}; \dots; \Gamma_n) \text{ if } \mathbf{c}(i) = \mathbf{1} \\
(\Gamma_1; \dots; \Gamma_{k-1}; \mathbf{c}(i); \Gamma_{k+1}; \dots; \Gamma_n) \xrightarrow{\perp_i} (\Gamma_1; \dots; \Gamma_{k-1}; \epsilon; \Gamma_{k+1}; \dots; \Gamma_n) \text{ if } \mathbf{c}(i) = \perp \\
(\Gamma_1; \dots; \Gamma_k, \mathbf{c}(i); \dots; \Gamma_n) \xrightarrow{\otimes_i} (\Gamma_1; \dots; \Gamma_k, A, B; \dots; \Gamma_n) \text{ if } \mathbf{c}(i) = A \otimes B \\
(\Gamma_1; \dots; \Gamma_k, \mathbf{c}(i); \dots; \Gamma_n) \xrightarrow{\wp_i} (\Gamma_1; \dots; \Gamma_k, A, B; \dots; \Gamma_n) \text{ if } \mathbf{c}(i) = A \wp B \\
(\Gamma_1; \dots; \Gamma_k, \mathbf{c}(i); \dots; \Gamma_n) \xrightarrow{?_i} (\Gamma_1; \dots; \Gamma_k, ?A, ?A; \dots; \Gamma_n) \text{ if } \mathbf{c}(i) = ?A \\
(\Gamma_1; \dots; \Gamma_k, \mathbf{c}(i); \dots; \Gamma_n) \xrightarrow{?_i} (\Gamma_1; \dots; \Gamma_k, A; \dots; \Gamma_n) \text{ if } \mathbf{c}(i) = ?A \\
(\Gamma_1; \dots; \Gamma_{k-1}; \mathbf{c}(i); \Gamma_{k+1}; \dots; \Gamma_n) \xrightarrow{?_i} (\Gamma_1; \dots; \Gamma_{k-1}; \epsilon; \Gamma_{k+1}; \dots; \Gamma_n) \text{ if } \mathbf{c}(i) = ?A \\
(\Gamma_1; \dots; \Gamma_{k-1}; ?A_1, \overset{p \geq 0}{?}, ?A_p, \mathbf{c}(i); \Gamma_{k+1}; \dots; \Gamma_n) \xrightarrow{\text{box}_i} (\Gamma_1; \dots; \Gamma_{k-1}; ?A_1, \overset{p \geq 0}{?}, ?A_p, A; \Gamma_{k+1}; \dots; \Gamma_n) \text{ if } \mathbf{c}(i) = !A
\end{array}$$

FIGURE 7. The generators of **Sched**. In the source Γ of each arrow, $\mathbf{c}(i)$ is the i^{th} formula in the flattening of Γ (the arrow's name keeps track of i). The Γ_k 's are (possibly empty) lists of MELL formulas.

i and j are not connected in the (undirected) graph $\|R\|$ of R , then i and j are not connected in the (undirected) graph $\|\rho\|$ of ρ . And if i and j are connected in the (undirected) graph of $\|\rho_{r_1 \dots r_n}\|$ of $\rho_{r_1 \dots r_n}$, then i and j are output tails of the same \boxtimes -cell in $\rho_{r_1 \dots r_n}$.

6. MEANS OF DESTRUCTION: UNWINDING MELL QUASI-PROOF-STRUCTURES

Our aim is to deconstruct (MELL or DiLL₀) proof-structures from their conclusions. To do that, we introduce the category of schedulings. The arrows of this category are sequences of deconstructing rules, acting on lists of lists of MELL formulas. These arrows act through functors on (MELL or DiLL₀) quasi-proof-structures, exhibiting their sequential structure.

Definition 6.1 (the category **Sched**). Let **Sched** be the category of schedulings whose

- objects are lists $\Gamma = (\Gamma_1; \dots; \Gamma_n)$ of lists of MELL formulas, with $n \geq 0$;
- arrows are freely generated by composition of the *elementary schedulings* in Figure 7; identities are the empty sequence of elementary schedulings, called *empty scheduling*.

A *scheduling* is any arrow $\nu: \Gamma \rightarrow \Gamma'$ in **Sched**. We write the composition of schedulings by juxtaposition in the diagrammatic order; so, if $\nu: \Gamma \rightarrow \Gamma'$ and $\nu': \Gamma' \rightarrow \Gamma''$, then $\nu\nu': \Gamma \rightarrow \Gamma''$. If $\nu = a_1 \cdots a_k$ where $k \geq 0$ and the a_i 's are elementary schedulings, the *length* of ν is k .

Reading Figure 7 left-to-right, mix_i is the only elementary scheduling that changes (and increases) the number of lists in a list Γ of lists of MELL formulas. The only elementary schedulings decreasing the number of MELL formulas in the flattening of Γ are $\text{ax}_i, \mathbf{1}_i, \perp_i, ?_{wi}$.

Example 6.2. $\wp_1 \wp_2 \wp_3 \otimes_1 \otimes_3 \text{exc}_1 \text{exc}_2 \text{mix}_2 \text{ax}_2 \text{exc}_2 \text{mix}_2 \text{ax}_2 \text{ax}_2$ is a scheduling from $((X \otimes Y^\perp) \wp ((Y \otimes Z^\perp) \wp (X^\perp \wp Z)))$ to 3ϵ , the list of 3 empty lists of MELL formulas.

The category **Sched** acts on MELL quasi-proof-structures, exhibiting a sequential structure in their construction, by means of the rewrite rules in Figure 8 (Definition 6.3). To ease their reading, we make explicit the name of the active conclusions (to be intended as positive

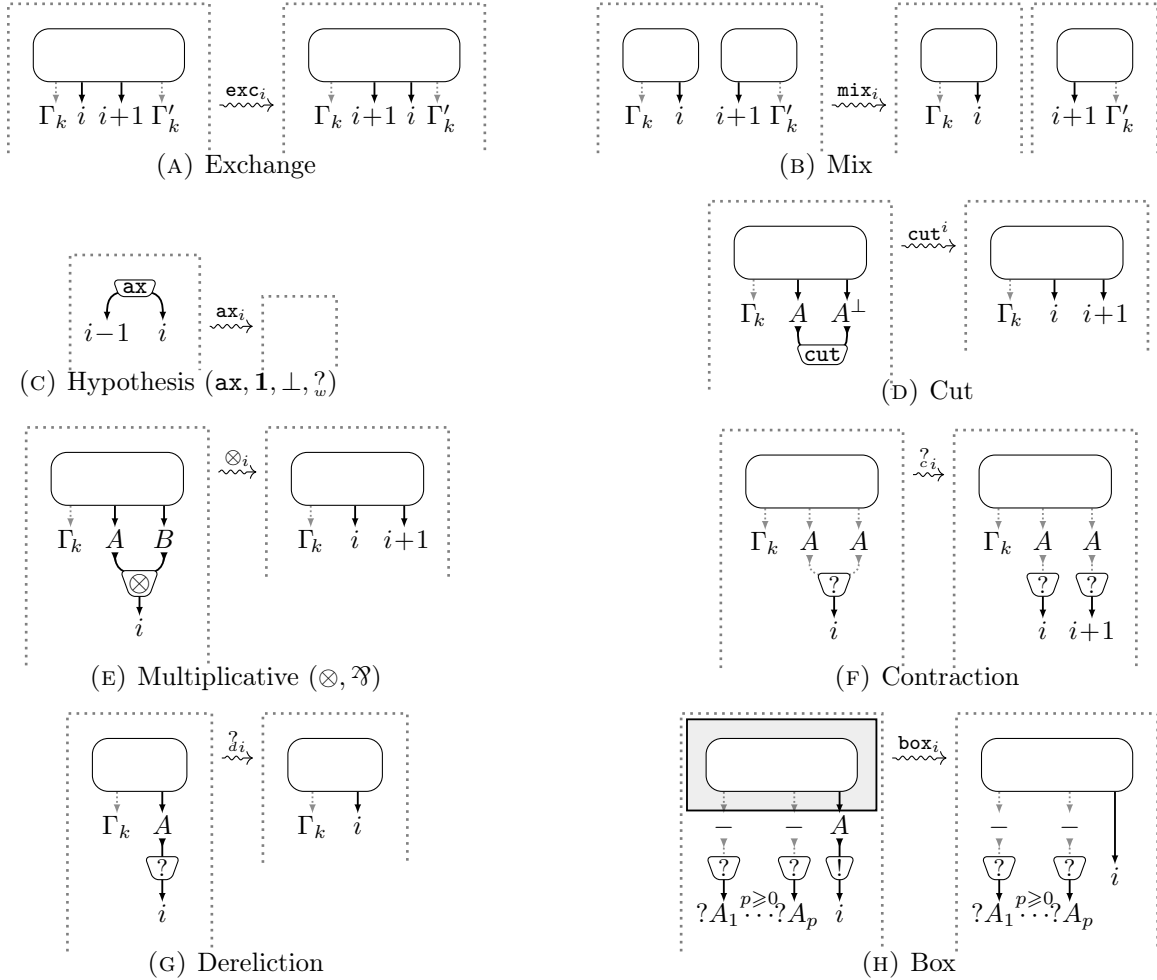


FIGURE 8. Action of elementary schedulings on MELL quasi-proof-structures.

integers), we only draw the components affected by the rewrite rule and omit the ones left unchanged; *e.g.*, if the affected component consists only of an \mathbf{ax} -cell whose outputs are the conclusions $i-1$ and i , we write $\begin{array}{c} \mathbf{ax} \\ \downarrow \downarrow \\ i-1 \quad i \end{array}$ ignoring the other components. Given a list Γ of lists of MELL formulas, $\mathbf{qMELL}(\Gamma)$ is the set of MELL quasi-proof-structures of type Γ .

Definition 6.3 (action of schedulings on MELL quasi-proof-structures). An elementary scheduling $a: \Gamma \rightarrow \Gamma'$ defines a relation $\overset{a}{\rightsquigarrow} \subseteq \mathbf{qMELL}(\Gamma) \times \mathbf{qMELL}(\Gamma')$, called the *action* of a , as the smallest relation containing all the cases in Figure 8, with the following remarks:

exchange: $\overset{\mathbf{exc}_i}{\rightsquigarrow}$ swaps the order of two consecutive conclusions in the same component.

mix: given a component R with at least two conclusions i and $i+1$ such that no conclusion $j \leq i$ is connected in $\|R\|$ to any conclusion $j' \geq i+1$, $\overset{\mathbf{mix}_i}{\rightsquigarrow}$ splits R into two components, one with the conclusions $j \leq i$, the other with the conclusions $j' \geq i+1$.²¹

hypothesis: if $a \in \{\mathbf{ax}_i, \mathbf{1}_i, \perp_i, ?_i\}$, $\overset{a}{\rightsquigarrow}$ deletes an hypothesis (*i.e.* a cell without inputs) that is the only cell in its component, yielding an empty component. We have drawn the

²¹See Footnote 25 for a more precise account. Read in reverse, \mathbf{mix}_i is analogous to the mix rule in the sequent calculus for MELL: it merges two components into one component putting together their conclusions.

- case **ax** in Figure 8c, the other cases (**1**, **⊥**, **?**) are similar, except for the number of conclusions.
- cut**: read in reverse, $\overset{\text{cut}}{\rightsquigarrow}^i$ relates a quasi-proof-structure with two conclusions i and $i + 1$ with the quasi-proof-structure where these two conclusions are cut by a **cut**-cell of depth 0. This rule, from left to right, is nondeterministic (as there are many possible cuts).
- multiplicative**: if $a \in \{\otimes_i, \wp_i\}$, $\overset{a}{\rightsquigarrow}$ deletes a cell labeled by a multiplicative connective, \otimes or \wp . The new conclusions i and $i + 1$ inherit the order from the inputs of the erased cell. We have only drawn the case \otimes in Figure 8e, the case \wp is similar.
- contraction**: $\overset{?}{\rightsquigarrow}^i$ splits a $?$ -cell with $h+k$ inputs into two $?$ -cells (so, it duplicates a $?$ -cell) with h and k inputs, respectively ($h, k \geq 0$). The rule, from left to right, is nondeterministic.
- dereliction**: $\overset{?}{\rightsquigarrow}^i$ only applies if the $?$ -cell (with 1 input) does not shift a level in the box-forest, *i.e.* it is not just outside a box (otherwise $\overset{?}{\rightsquigarrow}^i$ would not yield a MELL quasi-proof-structure).
- box**: $\overset{\text{box}_i}{\rightsquigarrow}$ only applies if a box (and its border) is alone in its component. In particular, each $?$ -cell represented in Figure 8h has at least one input (this is what “ $-$ ” stands for).

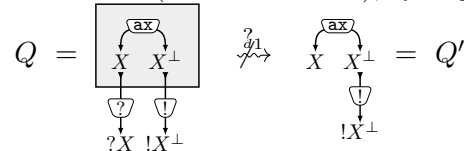
The rewrite relation is extended by composition of relations to define, for every scheduling $\nu: \Gamma \rightarrow \Gamma'$, a relation $\rightsquigarrow \subseteq \mathbf{qMELL}(\Gamma) \times \mathbf{qMELL}(\Gamma')$, called the *action* of ν .²²

Except **mix** _{i} , **exc** _{i} and $?_i$, the action of an elementary scheduling is a rewrite rule on MELL quasi-proof-structures that destroys either a **cut**-cell of depth 0 or a cell whose output is a conclusion. The types of the conclusions are updated according to Figure 7. The action of a scheduling is just the composition of a number (possibly none) of these rewrite rules.

The action of schedulings cannot decrease the number of components of a MELL quasi-proof-structure, and the action of **mix** _{i} is the only one that increases this number.

Among the actions of elementary schedulings in Figure 8, the cases **mix** and **box** are the only ones that change the box-forest of a quasi-proof-structure: **mix** splits a tree in two distinct trees by splitting its root, while **box** merges a root of a tree with a non-root vertex just above it. For instance, the action **mix** _{i} rewrites the MELL proof-structure S in Figure 3 to the MELL quasi-proof-structure R in Figure 4, when i is the conclusion of S of type $?!1$.

The way the action of an elementary scheduling is defined (Definition 6.3) automatically rules out the possibility that a MELL quasi-proof-structure rewrites to a module that is not a MELL quasi-proof-structure. For instance, the elementary scheduling $?_1$ cannot act on the MELL proof-structure Q below, because it would yield a module Q' that is not a MELL quasi-proof-structure, as in Q' it would be impossible to define a box-function that fulfills the constraints of Definition 4.1 for MELL (see Footnote 15); Q' is just a DiLL_0 proof-structure.



Actions of schedulings can be seen as arrows in the category **Rel** of sets and relations.

Definition 6.4 (functor **qMELL**). We define a functor **qMELL**: **Sched** \rightarrow **Rel** by:

²²That is, given the schedulings $\nu: \Gamma \rightarrow \Gamma'$ and $\nu': \Gamma' \rightarrow \Gamma''$ and the quasi-proof-structures $R \in \mathbf{qMELL}(\Gamma)$ and $R'' \in \mathbf{qMELL}(\Gamma'')$, $R \rightsquigarrow \nu' R''$ if and only if there is $R' \in \mathbf{qMELL}(\Gamma')$ such that $R \rightsquigarrow \nu R'$ and $R' \rightsquigarrow \nu' R''$ (often denoted by $R \rightsquigarrow \nu' R''$). The action of the empty scheduling is the identity relation.

- *on objects*: for every list Γ of lists of MELL formulas, $\mathbf{qMELL}(\Gamma)$ is the set of MELL quasi-proof-structures of type Γ ;
- *on arrows*: for any $\nu: \Gamma \rightarrow \Gamma'$, $\mathbf{qMELL}(\nu)$ is $\mathcal{L}_\nu: \mathbf{qMELL}(\Gamma) \rightarrow \mathbf{qMELL}(\Gamma')$ (Definition 6.3).

Lemmas 6.5, 6.6, 6.7 and Proposition 6.8 express some properties of our rewrite rules.

Lemma 6.5 (co-functionality). *Let $\nu: \Gamma \rightarrow \Gamma'$ be a scheduling. The action \mathcal{L}_ν of ν is a co-function from $\mathbf{qMELL}(\Gamma)$ to $\mathbf{qMELL}(\Gamma')$, that is, a function $\mathcal{L}_\nu^{\text{op}}: \mathbf{qMELL}(\Gamma') \rightarrow \mathbf{qMELL}(\Gamma)$.*

Let R, R' be MELL quasi-proof-structures, a be an elementary scheduling. We say that:

- a *applies* to R if $R \xrightarrow{\text{exc}_{i_1} \cdots \text{exc}_{i_n} a} R'$ for some elementary schedulings $\text{exc}_{i_1}, \dots, \text{exc}_{i_n}$ ($n \geq 0$);
- $R \xrightarrow{a} R'$ is *nullary $?_c$ -splitting* if $a = ?_c$ and its action from R to R' splits the $?_c$ -cell of R with output i and $h \geq 0$ inputs into two $?_c$ -cells, one with h inputs and one with 0 inputs.

Lemma 6.6 (applicability of elementary scheduling). *Let R be a non-empty MELL quasi-proof-structure. Then an elementary scheduling $a \in \{\text{mix}_i, \text{ax}_i, \mathbf{1}_i, \perp_i, \otimes_i, \mathfrak{A}_i, ?_c, \mathfrak{A}_i, \mathfrak{A}_i, \mathfrak{A}_i, \text{cut}^i, \text{box}_i\}$ applies to R , for some conclusion i of R (or input i of a cut-cell of depth 0 in R).*

No elementary scheduling applies to an empty quasi-proof-structure.

To deconstruct a MELL quasi-proof-structure R , nullary $?_c$ -splitting actions are problematic because they increase any reasonable size for R . Luckily, the lemma below says roughly that, in MELL, nullary $?_c$ -splitting actions are superfluous as means of destruction. Remark 9.6 will show that they are needed to achieve the naturality result (Theorem 7.5).

Lemma 6.7 (splitting). *Let R be a MELL quasi-proof-structure with at least a conclusion i . If $a = ?_c$ and $R \xrightarrow{a} R'$ is nullary $?_c$ -splitting, then for some MELL quasi-proof-structure S :*

- (1) *either i is the output of a $?_c$ -cell of R with $k \geq 2$ inputs and $R \xrightarrow{a} S$ is not nullary $?_c$ -splitting;*
- (2) *or i is the output of a $?_c$ -cell of R with 0/1 inputs and in the same component as i there are conclusions i_0, \dots, i_n with $n \geq 0$ (possibly $i = i_j$ or $i_j = i_{j'}$ for some $0 \leq j, j' \leq n$; i_0 may also be an input of a cut-cell of depth 0) such that $R \xrightarrow{\text{exc}_{i_1} \cdots \text{exc}_{i_n}} S' \xrightarrow{a'} S$ with $a' \in \{\text{mix}_{i_0}, \text{box}_{i_0}, \otimes_{i_0}, \mathfrak{A}_{i_0}, \text{cut}^{i_0}, ?_{d_{i_0}}, ?_{w_{i_0}}, ?_{c_{i_0}}\}$ and if $a' = ?_{c_{i_0}}$ then $S' \xrightarrow{a'} S$ is not nullary $?_c$ -splitting.*

Lemmas 6.5, 6.6 and 6.7 are proven by simple inspection of the rewrite rules in Figure 8. The proof of Lemma 6.5 has a subtle case: given $R' \in \mathbf{qMELL}(\Gamma')$, if $\nu = ?_c$ there is a (unique) $R \in \mathbf{qMELL}(\Gamma)$ such that $R \xrightarrow{\mathcal{L}_\nu} R'$ because any conclusion of R' of type $?A$ is the output of a $?_c$ -cell (thanks to atomic axioms, Footnote 9; for non-atomic axioms, see Section 10).

Given a scheduling ν , we say $R \xrightarrow{\mathcal{L}_\nu} R'$ *has nullary $?_c$ -splittings* if there is an elementary scheduling a such that $R \xrightarrow{\mathcal{L}_\nu} R' = R \xrightarrow{\nu_1} R_1 \xrightarrow{a} R_2 \xrightarrow{\nu_2} R'$ and $R_1 \xrightarrow{a} R_2$ is nullary $?_c$ -splitting.

Every MELL quasi-proof-structure can be unwound from its conclusions to an empty one by the action of some scheduling, and nullary $?_c$ -splittings are not needed for that.

Proposition 6.8 (normalization). *Let R be a MELL quasi-proof-structure of type Γ . Then, there exists a scheduling $\nu: \Gamma \rightarrow \vec{c}$ such that $R \xrightarrow{\mathcal{L}_\nu} \vec{c}$, without nullary $?_c$ -splittings.*

Proof. Let the *size* of a MELL quasi-proof-structure R be the triple (p, q, r) where:

- p is the (finite) multiset of the number of inputs of each $?_c$ -cell in R ;
- q is the number of cells in R ;
- r is the (finite) multiset of the number of conclusions of each component of R .

Multisets are well-ordered as usual, triples are well-ordered lexicographically.

By Lemmas 6.6 and 6.7, it suffices to show that such a size is left unchanged by $\overset{\text{exc}_i}{\rightsquigarrow}$ and decreases with any action of $\overset{?}{c}_i$ that is not nullary $\overset{?}{c}$ -splitting (p decreases) and with any action in Figure 8 other than the ones of exc_i and $\overset{?}{c}_i$ (the action of mix_i leaves p and q unchanged and decreases r ; the remaining actions leave p unchanged and decrease q). \square

Proposition 6.8 and the last statement in Lemma 6.6 imply that actions of elementary schedulings define a weakly normalizing rewrite system on MELL quasi-proof-structures: the normal forms are exactly the empty quasi-proof-structures $n\varepsilon$, for all $n > 0$. This rewrite system is not strongly normalizing because of the actions of exc_i and, more significantly, $\overset{?}{c}_i$ when are nullary $\overset{?}{c}$ -splitting. An example of an indefinitely extendable rewriting is below.

$$\begin{array}{ccccccc} \begin{array}{c} \text{?} \\ \downarrow \\ \text{?1} \end{array} & \overset{?c_1}{\rightsquigarrow} & \begin{array}{c} \text{?} \\ \downarrow \\ \text{?1} \end{array} & \begin{array}{c} \text{?} \\ \downarrow \\ \text{?1} \end{array} & \overset{?c_1}{\rightsquigarrow} & \begin{array}{c} \text{?} \\ \downarrow \\ \text{?1} \end{array} & \begin{array}{c} \text{?} \\ \downarrow \\ \text{?1} \end{array} & \begin{array}{c} \text{?} \\ \downarrow \\ \text{?1} \end{array} & \overset{?c_1}{\rightsquigarrow} & \dots \end{array}$$

The action of a scheduling ν such that $R \overset{\nu}{\rightsquigarrow} \varepsilon$, read backward, can be seen as a sequence of elementary steps to build the MELL quasi-proof-structure R from an empty one ε . An example of a rewriting from some R to ε is in Figure 11 (Section 8). Intuitively, a scheduling $\nu: (\Gamma) \rightarrow \varepsilon$ “encodes” a way to build a MELL proof-structure of type Γ from scratch.

7. NATURALITY OF UNWINDING DiLL₀ QUASI-PROOF-STRUCTURES

We show here how the actions of schedulings act on DiLL₀ quasi-proof-structures, mimicking the behavior of the actions of schedulings on MELL quasi-proof-structures seen in Section 6. Schedulings are the same, the novelty is that DiLL₀ quasi-proof-structures have no boxes and might have daimons. The bridge between the MELL and DiLL₀ frameworks is given by the filled Taylor expansion, which actually defines a *natural transformation* (Theorem 7.5).

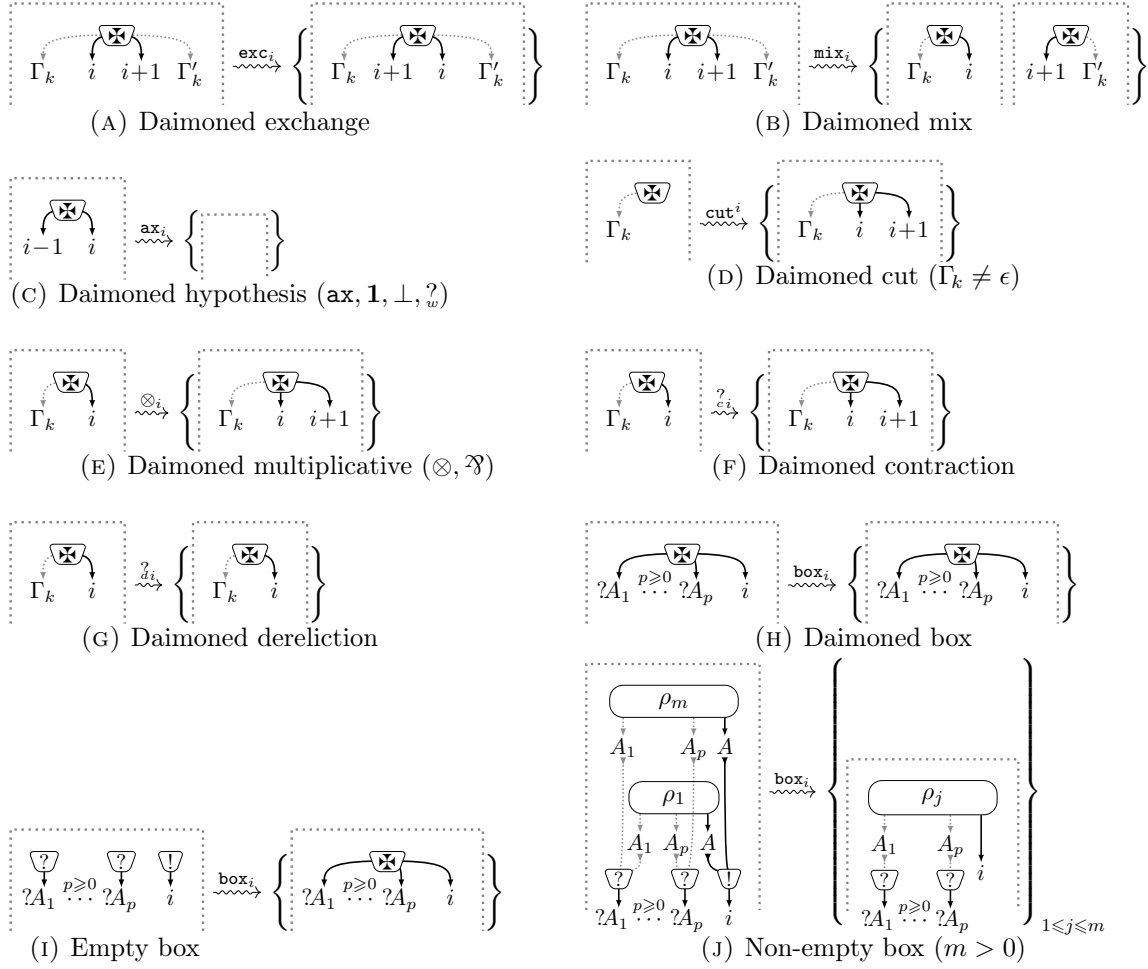
For Γ a list of lists of MELL formulas, $\text{qDiLL}_0(\Gamma)$ is the set of DiLL₀ quasi-proof-structures of type Γ (we assume they have the *same* conclusions). For any set X , its powerset is $\mathfrak{P}(X)$.

Definition 7.1 (action of schedulings on DiLL₀ quasi-proof-structures). An elementary scheduling $a: \Gamma \rightarrow \Gamma'$ defines a relation $\overset{a}{\rightsquigarrow} \subseteq \text{qDiLL}_0(\Gamma) \times \mathfrak{P}(\text{qDiLL}_0(\Gamma'))$, called the *action* of a , as the smallest relation containing all the rules in Figure 8—except Figure 8h, and with the same remarks²³—and in Figure 9. We extend it to a relation $\overset{a}{\rightsquigarrow} \subseteq \mathfrak{P}(\text{qDiLL}_0(\Gamma)) \times \mathfrak{P}(\text{qDiLL}_0(\Gamma'))$ by the multiplication of the monad powerset $X \mapsto \mathfrak{P}(X)$.²⁴ The rewrite relation on $\mathfrak{P}(\text{qDiLL}_0(\Gamma)) \times \mathfrak{P}(\text{qDiLL}_0(\Gamma'))$ is extended by composition of relations to define, for any scheduling $\nu: \Gamma \rightarrow \Gamma'$, a relation $\overset{\nu}{\rightsquigarrow} \subseteq \mathfrak{P}(\text{qDiLL}_0(\Gamma)) \times \mathfrak{P}(\text{qDiLL}_0(\Gamma'))$, the *action* of ν .

Roughly, any rewrite rule in Figure 9—except Figures 9i and 9j—mimics the behavior of the corresponding rule in Figure 8 using a \boxtimes -cell. Since every elementary scheduling has at least two versions of its action, one in Figure 8 and one (or two for box_i) in Figure 9, when $\Pi \overset{a}{\rightsquigarrow} \Pi'$, some elements of Π may rewrite according to one version of the action of a , others elements according to another version. In any case, all the elements of Π' have the same conclusions, as the elements of Π do. Types are updated according to Figure 7.

²³More precisely, if $\overset{a}{\rightsquigarrow} \subseteq \text{qDiLL}_0(\Gamma) \times \mathfrak{P}(\text{qDiLL}_0(\Gamma'))$, in the rewrite rule $\overset{a}{\rightsquigarrow}$ in Figure 8 the quasi-proof-structure on the right-hand-side must be replaced by the singleton set containing it.

²⁴More explicitly, given $\Pi \subseteq \text{qDiLL}_0(\Gamma)$ and $\Pi' \subseteq \text{qDiLL}_0(\Gamma')$, $\Pi \overset{a}{\rightsquigarrow} \Pi'$ means that $\Pi' = \bigcup \{\rho' \subseteq \text{qDiLL}_0(\Gamma') \mid \exists \rho \in \Pi : \rho \overset{a}{\rightsquigarrow} \rho'\}$. Thus, $\rho \overset{a}{\rightsquigarrow} \rho' \subseteq \Pi'$ for every $\rho \in \Pi$. And, in particular, $\emptyset \overset{a}{\rightsquigarrow} \emptyset$. Roughly, the same rewrite rule applies to the *same* conclusion of *each* element of a set of DiLL₀ quasi-proof-structures. Note that $\emptyset \neq \Pi \overset{a}{\rightsquigarrow} \Pi'$ implies $\Pi' \neq \emptyset$, since there is no elementary scheduling a such that $\rho \overset{a}{\rightsquigarrow} \emptyset$.

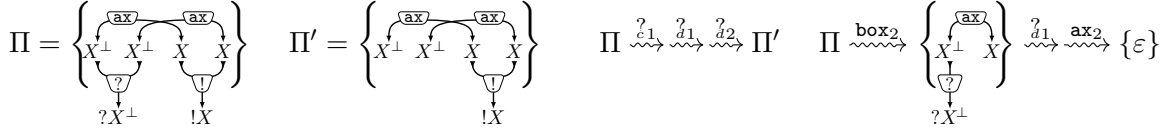


Lists Γ_k and $\Gamma_{k'}$ may be empty, except in Figure 9d since a \boxtimes -cell has at least an output.

FIGURE 9. Action of elementary schedulings on DiLL_0 quasi-proof-structures.

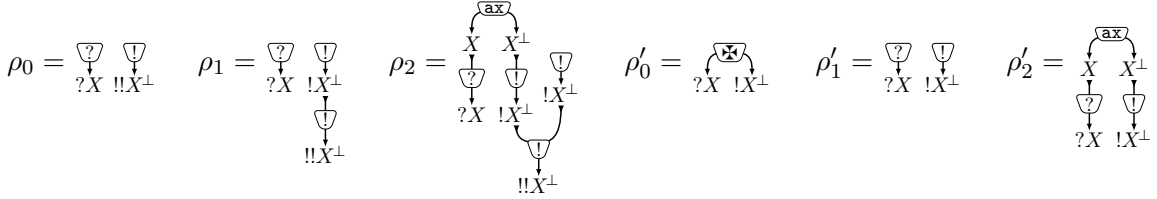
The action in Figure 9i acts on a component made of one *co-weakening* (a $!$ -cell without inputs) and some *weakenings* ($?$ -cells without inputs), it erases them and creates a \boxtimes -cell, whose outputs are all the conclusions of the component. If we just erased the co-weakening with output of type $!A$, we would lose the information about the type A ; this is one of the reasons why a \boxtimes -cell is needed. Intuitively, a co-weakening represents a box taken 0 times, so there is no information about the content of the box, apart from its type. The daimon is a “universal” DiLL_0 proof-structure representing this information plainly missing.

The action in Figure 9j separates the copies of a candidate for a box. It requires that, in the graph of the component on the left of $\overset{\text{box}_i}{\rightsquigarrow}$, ρ_j is not connected to $\rho_{j'}$ for $j \neq j'$, except for the $!$ -cell and the $?$ -cells in the conclusions. Read in reverse, the rule associates with a non-empty finite set of DiLL_0 quasi-proof-structures $\{\rho_1, \dots, \rho_m\}$ the merging of ρ_1, \dots, ρ_m , that is the DiLL_0 quasi-proof-structure depicted on the left of $\overset{\text{box}_i}{\rightsquigarrow}$. Intuitively, ρ_1, \dots, ρ_m represent $m > 0$ possible copies of a box, whose “compatibility” must be analyzed “in parallel”: this is why the action rewrites a single DiLL_0 quasi-proof-structure to a (finite) *set* of them.

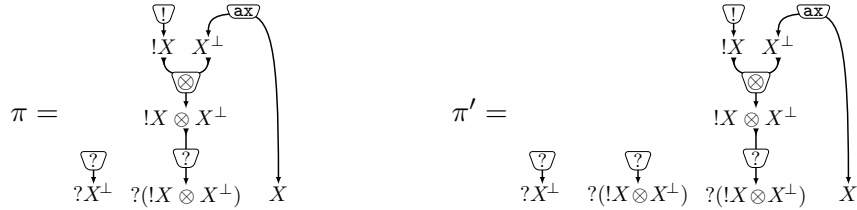

 FIGURE 10. Examples of actions of schedulings for DiLL_0 .

Note that a \boxtimes -cell can be created only by the rewrite rule in Figure 9i, and erased only by the action of elementary scheduling called hypothesis in Figure 9c: $\text{ax}_i, \mathbf{1}_i, \perp_i, \mathbf{?}_i$.

Example 7.2. Consider the DiLL_0 proof-structures below (X is a propositional variable). We have that $\{\rho_0, \rho_1, \rho_2\} \xrightarrow{\text{box}_2} \{\rho'_0, \rho'_1, \rho'_2\}$ because $\rho_0 \xrightarrow{\text{box}_2} \{\rho'_0\}$ (according to Figure 9i), $\rho_1 \xrightarrow{\text{box}_2} \{\rho'_1\}$ (according to Figure 9j) and $\rho_2 \xrightarrow{\text{box}_2} \{\rho'_2\}$ (according to Figure 9j).



Example 7.3. Consider the DiLL_0 proof-structures π and π' below.



For every $n > 0$, the singleton $\{\pi\}$ rewrites to $\{(n+1)\varepsilon\}$ by the action of the scheduling ν_n^1 where, for all $k > 0$, $\nu_n^k: (\mathbf{?}_X^\perp, \mathbf{?}(\mathbf{!}X \otimes X^\perp), X; \varepsilon; \dots; \varepsilon) \rightarrow (k+n)\varepsilon$ is defined by:

$$\nu_1^k = \mathbf{?}_2 \otimes_2 \text{mix}_2 \text{ax}_4 \text{box}_2 \mathbf{?}_1 \text{ax}_2 \quad \nu_{n+1}^k = \mathbf{?}_2 \mathbf{?}_3 \otimes_3 \text{mix}_3 \text{ax}_5 \text{box}_3 \nu_n^{k+1}.$$

Indeed, we have the following rewritings, where $\pi_{\boxtimes} = \mathbf{?}_X^\perp \mathbf{?}(\mathbf{!}X \otimes X^\perp) X$

$$\{\pi_{\boxtimes}\} \xrightarrow{\mathbf{?}_2} \otimes_2 \text{mix}_2 \text{ax}_4 \text{box}_2 \mathbf{?}_1 \text{ax}_2 \{2\varepsilon\} \quad \{\pi_{\boxtimes}\} \xrightarrow{\mathbf{?}_2} \mathbf{?}_3 \otimes_3 \text{mix}_3 \text{ax}_5 \text{box}_3 \left\{ \mathbf{?}_X^\perp \mathbf{?}(\mathbf{!}X \otimes X^\perp) X \right\}$$

and therefore, for every $n > 1$, the scheduling ν_n^1 acts on $\{\pi\}$ in the way below

$$\{\pi\} \xrightarrow{\mathbf{?}_2} \{\pi'\} \xrightarrow{\mathbf{?}_3} \otimes_3 \text{mix}_3 \text{ax}_5 \left\{ \mathbf{?}_X^\perp \mathbf{?}(\mathbf{!}X \otimes X^\perp) \mathbf{!}X \right\} \xrightarrow{\text{box}_3} \left\{ \mathbf{?}_X^\perp \mathbf{?}(\mathbf{!}X \otimes X^\perp) X \right\} \xrightarrow{\nu_{n-1}^2} \{(n+1)\varepsilon\}.$$

The rewriting on DiLL_0 quasi-proof-structures behaves differently than on MELL quasi-proof-structures. In DiLL_0 , Lemma 6.6 and Proposition 6.8 are false. Indeed, in Figure 10 no elementary scheduling applies to the singleton Π' of a non-empty DiLL_0 proof-structure. So, $\Pi \xrightarrow{\mathbf{?}_1} \mathbf{?}_1 \mathbf{?}_2 \mathbf{?}_2 \Pi'$ and gets stuck, but the action of another scheduling rewrites Π to $\{\varepsilon\}$ (Figure 10 on the right). Thus, some rewritings from Π end in $\{\varepsilon\}$, others get stuck before.

Definition 7.4 (functor $\mathfrak{P}qDiLL_0$). We define a functor $\mathfrak{P}qDiLL_0: \mathbf{Sched} \rightarrow \mathbf{Rel}$ by:

- *on objects*: for every list Γ of lists of MELL formulas, $\mathfrak{P}qDiLL_0(\Gamma) = \mathfrak{P}(qDiLL_0(\Gamma))$, the powerset of the set $qDiLL_0(\Gamma)$ of $DiLL_0$ quasi-proof-structures of type Γ ;
- *on arrows*: for any $\nu: \Gamma \rightarrow \Gamma'$, $\mathfrak{P}qDiLL_0(\nu)$ is $\rightsquigarrow: \mathfrak{P}qDiLL_0(\Gamma) \rightarrow \mathfrak{P}qDiLL_0(\Gamma')$ (Def. 7.1).

We can now compare the functors $qMELL$ and $\mathfrak{P}qDiLL_0$ from \mathbf{Sched} to \mathbf{Rel} .

Theorem 7.5 (naturality). *The filled Taylor expansion defines a natural transformation*

$$\mathfrak{T}^{\mathfrak{X}}: \mathfrak{P}qDiLL_0 \Rightarrow qMELL: \mathbf{Sched} \rightarrow \mathbf{Rel}$$

by: for Γ a list of lists of MELL formulas, $(\Pi, R) \in \mathfrak{T}^{\mathfrak{X}}_\Gamma$ iff $\Pi \subseteq \mathcal{T}^{\mathfrak{X}}(R)$ and the type of R is Γ .

In other words, diagram (7.1) below commutes for every scheduling $\nu: \Gamma \rightarrow \Gamma'$.

$$\begin{array}{ccc} \mathfrak{P}qDiLL_0(\Gamma) & \xrightarrow{\mathfrak{P}qDiLL_0(\nu)} & \mathfrak{P}qDiLL_0(\Gamma') \\ \mathfrak{T}^{\mathfrak{X}}_\Gamma \downarrow & & \downarrow \mathfrak{T}^{\mathfrak{X}}_{\Gamma'} \\ qMELL(\Gamma) & \xrightarrow{qMELL(\nu)} & qMELL(\Gamma') \end{array} \quad (7.1) \quad \begin{array}{ccc} \Pi & \rightsquigarrow & \Pi' \\ \mathfrak{T}^{\mathfrak{X}}_\Gamma \downarrow & & \downarrow \mathfrak{T}^{\mathfrak{X}}_{\Gamma'} \\ R & \rightsquigarrow & R' \end{array} \quad (7.2) \quad \begin{array}{ccc} \Pi & \rightsquigarrow & \Pi' \\ \mathfrak{T}^{\mathfrak{X}}_\Gamma \downarrow & & \downarrow \mathfrak{T}^{\mathfrak{X}}_{\Gamma'} \\ R & \rightsquigarrow & R' \end{array} \quad (7.3)$$

That is, given $\Pi \rightsquigarrow \Pi'$ with $\Pi' \subseteq \mathcal{T}^{\mathfrak{X}}(R')$, we can *simulate backward* the rewriting to R (here the co-functionality of actions on $qMELL$ expressed by Lemma 6.5 comes in handy) so that $R \rightsquigarrow R'$ and $\Pi \subseteq \mathcal{T}^{\mathfrak{X}}(R)$ (square (7.2)); and conversely, given $R \rightsquigarrow R'$, we can *simulate forward* the rewriting for any $\Pi \subseteq \mathcal{T}^{\mathfrak{X}}(R)$, so that $\Pi \rightsquigarrow \Pi'$ for some $\Pi' \subseteq \mathcal{T}^{\mathfrak{X}}(R')$ (square (7.3)). In both squares, the *same* kind of action is performed on $DiLL_0$ and on $MELL$.

Proof. $\mathfrak{T}^{\mathfrak{X}}$ is a family of arrows of \mathbf{Rel} indexed by the objects in \mathbf{Sched} . We have to show that squares (7.2)–(7.3) commute for any scheduling $\nu: \Gamma \rightarrow \Gamma'$, and actually, it is enough to show commutation for elementary schedulings. Let $a: \Gamma \rightarrow \Gamma'$ be an elementary scheduling.

(1) *Square (7.3)*: Let us prove that $qMELL(a) \circ \mathfrak{T}^{\mathfrak{X}}_\Gamma \subseteq \mathfrak{T}^{\mathfrak{X}}_{\Gamma'} \circ \mathfrak{P}qDiLL_0(a)$.

Let $(\Pi, R') \in qMELL(a) \circ \mathfrak{T}^{\mathfrak{X}}_\Gamma$. Let $R = (|R|, \mathcal{F}, \text{box}) \in qMELL(\Gamma)$ be a witness of composition, that is, an element such that $(\Pi, R) \in \mathfrak{T}^{\mathfrak{X}}_\Gamma$ and $R \rightsquigarrow R'$. If $\Pi = \emptyset$ then we are done because $\Pi \rightsquigarrow \emptyset$ (see Footnote 24) and $\emptyset \subseteq \mathcal{T}^{\mathfrak{X}}(R')$. So, we suppose that $\Pi \neq \emptyset$.

Let (t, h) be a thick subforest of \mathcal{F} , let r_1, \dots, r_n be some roots in \mathcal{F} (and in t , by Remark 5.3), and let $\rho_{r_1 \dots r_n} \in \Pi$ be the element of the filled Taylor expansion $\mathcal{T}^{\mathfrak{X}}(R)$ of R associated with (t, h) and r_1, \dots, r_n (i.e. $\rho_{r_1 \dots r_n}$ is the emptying on r_1, \dots, r_n of the element of $\mathcal{T}(R)$ obtained via the thick subforest (t, h) of \mathcal{F} , see Definition 5.8). By Remark 5.9, we can identify the conclusions of R and of $\rho_{r_1 \dots r_n}$. The case $a = \text{exc}_i$ is trivial since it just swaps the order of two consecutive conclusions. Other cases for a :

- If $a = \text{mix}_i$, let r be the root in \mathcal{F} corresponding to the conclusion i (i.e. $\text{box}_F(i)$ is the output of r) and let $p, \dots, p+k$ be the conclusions of the component S of R containing i . In the undirected graph $\|S\|$, none of the conclusions p, \dots, i is connected to any conclusion $i+1, \dots, p+k$. By Definitions 5.5 and 5.8, as $\rho_{r_1 \dots r_n} \in \Pi \subseteq \mathcal{T}^{\mathfrak{X}}(R)$, r is a root in the box-forest of $\rho_{r_1 \dots r_n}$, and $p, \dots, p+k$ are the conclusions of $\rho_{r_1 \dots r_n}$ corresponding to r . There are two cases, according to Remark 5.11:
 - the connected components of i and $i+1$ are disjoint in the undirected graph $\|\rho_{r_1 \dots r_n}\|$;
 - i and $i+1$ belong to the same connected component of $\|\rho_{r_1 \dots r_n}\|$, and their component in $\rho_{r_1 \dots r_n}$ is a daimon with conclusions $p, \dots, i, i+1, \dots, p+k$.

In both cases the rule mix_i is also applicable to $\rho_{r_1 \dots r_n}$, yielding a $DiLL_0$ quasi-proof-structure ρ' . The box-forest \mathcal{F}' of R' is obtained from the box-forest \mathcal{F} of R by

root-splitting²⁵the tree with root r in two trees with roots r'_1, r'_2 . Let t' be a forest obtained from t in a similar way. Let $h': t' \rightarrow \mathcal{F}'$ be the directed graph morphism such that $h'_V(r'_1) = r'_1, h'_V(r'_2) = r'_2$ and $h'_V(v) = h_V(v)$ for all $v \in V_{\mathcal{F}} \cap V_{\mathcal{F}'}$. We verify that ρ' is the element of $\mathcal{T}^{\mathbf{x}}(R')$ associated with the thick subforest (t', h') of \mathcal{F}' .

- If $a \in \{\otimes_i, \wp_i, ?_{di}, ?_{ci}, ?_{wi}, \mathbf{1}_i, \perp_i, \mathbf{ax}_i\}$, let k be such that the rule a acts on the conclusions $i - k, \dots, i$ of a component of R , and let ℓ be the type of the cell in R whose output is the conclusion i (or whose outputs are the conclusions $i - 1$ and i , if $\ell = \mathbf{ax}$). In $\rho_{r_1 \dots r_n}$ either there is a ℓ -cell with $\ell \neq \mathbf{ax}$ whose output is the conclusion i , or there is a \mathbf{ax} -cell whose outputs are the conclusions $i - 1$ and i , or the component containing the conclusion i is a daimon with conclusions $i - k, \dots, i$. Clearly a is applicable to $\rho_{r_1 \dots r_n}$, which yields a DiLL_0 quasi-proof-structure ρ' .

The box-forest \mathcal{F}' of R' is \mathcal{F} , the one of R . We verify that ρ' is the element of the filled Taylor expansion $\mathcal{T}^{\mathbf{x}}(R')$ of R' associated with the thick subforest (t, h) of \mathcal{F}' .

- If $a = \text{cut}^i$, let c be the cut-cell of depth 0 in R to which a is applied. As in the case above, the cut-cell c has either one image in $\rho_{r_1 \dots r_n}$ or is represented by a \mathbf{x} -cell (with at least one conclusion). In both cases, cut^i is applicable to $\rho_{r_1 \dots r_n}$, yielding ρ' .

The box-forest \mathcal{F}' of R' is \mathcal{F} , the one of R . We verify that ρ' is the element of the filled Taylor expansion $\mathcal{T}^{\mathbf{x}}(R')$ of R' associated with the thick subforest (t, h) of \mathcal{F}' .

- If $a = \mathbf{box}_i$, consider the component of R whose conclusions are $i - k, \dots, i$ with $k \geq 0$. In the component π of $\rho_{r_1 \dots r_n}$ with conclusions $i - k, \dots, i$, there are three cases:
 - (a) π is a daimon whose conclusions are $i - k, \dots, i$; so, the rule in Figure 9h applies;
 - (b) π consists of a $!$ -cell with output tail i and no inputs; and of k $?$ -cells without inputs; therefore, the rewrite rule in Figure 9i applies;
 - (c) π has a $!$ -cell c with output tail i and at least an input, k $?$ -cells c_1, \dots, c_k immediately above each of the other k conclusions, and other cells that can be split in m pairwise disjoint sub-graphs of $\|\pi'\|$, where $m \geq 1$ and π' is π without c, c_1, \dots, c_k and their output; therefore, the rewrite rule in Figure 9j applies.

In any case, \mathbf{box}_i applies to $\rho_{r_1 \dots r_n}$, yielding a family ρ'_1, \dots, ρ'_m of DiLL_0 proof-structures, where $m = 1$ in cases (a) and (b), while $m \geq 1$ in case (c).

The box-forest \mathcal{F}' of R' is obtained from the box-forest \mathcal{F} of R by erasing the root b corresponding to the conclusions $i - k, \dots, i$ (i.e., $\mathbf{box}_F(i - k) = \dots = \mathbf{box}_F(i)$ is the output of b): the new root b' of this tree of \mathcal{F}' is the unique vertex immediately above b in \mathcal{F} . We have $h^{-1}(b') = \{b'_1, \dots, b'_m\}$, and m trees t'_1, \dots, t'_m , where b'_j is the root of t'_j . The directed graph morphisms $h'_j: t'_j \rightarrow \mathcal{F}'$ are defined accordingly. We verify that ρ'_j is the element of $\mathcal{T}^{\mathbf{x}}(R')$ associated with the thick subforest (t'_j, h'_j) of \mathcal{F}' .

- (2) *Square (7.2)*: Let us prove that $\mathfrak{T}_{\Gamma'}^{\mathbf{x}} \circ \mathfrak{PqDiLL}_0(a) \subseteq \mathfrak{qMELL}(a) \circ \mathfrak{T}_{\Gamma}^{\mathbf{x}}$.

Let $(\Pi, R') \in \mathfrak{T}_{\Gamma'}^{\mathbf{x}} \circ \mathfrak{PqDiLL}_0(a)$ with $R' = (|R'|, \mathcal{F}', \mathbf{box}')$. Let Π' be a witness of composition, i.e. a set in $\mathfrak{PqDiLL}_0(\Gamma')$ such that $\Pi \xrightarrow{a} \Pi'$ and $(\Pi', R') \in \mathfrak{T}_{\Gamma}^{\mathbf{x}}$.

We want to exhibit a MELL quasi-proof-structure R such that $R \xrightarrow{a} R'$ and Π is a part of the filled Taylor expansion $\mathcal{T}^{\mathbf{x}}(R)$ of R . By co-functionality of $\mathfrak{qMELL}(a)$ (Lemma 6.5), we have a candidate for such an R : the preimage of R' by this co-functional relation \xrightarrow{a} . In other terms, if $a: \Gamma \rightarrow \Gamma'$ and $R' \in \mathfrak{qMELL}(\Gamma')$, then there exists a unique $R = (|R|, \mathcal{F}, \mathbf{box}) \in \mathfrak{qMELL}(\Gamma)$ such that $R \xrightarrow{a} R'$ (Lemma 6.5). We only have to check that $\Pi \subseteq \mathcal{T}^{\mathbf{x}}(R)$. If $\Pi = \emptyset$ then trivially $\Pi \subseteq \mathcal{T}^{\mathbf{x}}(R)$. So, we suppose that $\Pi \neq \emptyset$.

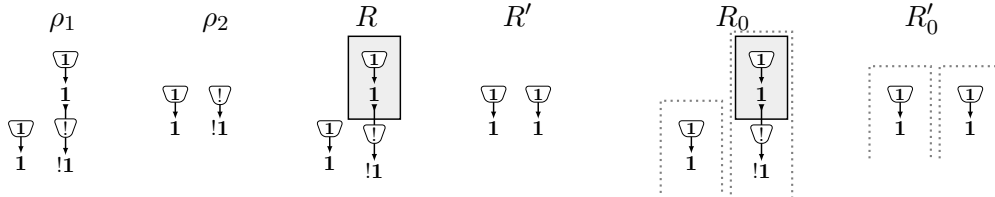
²⁵It means that $\mathcal{F} = \mathcal{F}'$ except that in \mathcal{F}' the root r with its output is replaced by the roots r'_1 and r'_2 with their output, and that the inputs of r are split into the inputs of r'_1 and the inputs of r'_2 .

Let $\rho \in \Pi$ and $\{\rho'_1, \dots, \rho'_m\} \subseteq \Pi'$ such that $\rho \rightsquigarrow \{\rho'_1, \dots, \rho'_m\}$. In all cases except $a = \mathbf{box}_i$, we have $m = 1$ *i.e.* $\{\rho'_1, \dots, \rho'_m\} = \{\rho'_1\}$. For $a \neq \mathbf{box}_i$, let (t', h') be a thick subforest of \mathcal{F}' , let r'_1, \dots, r'_k be some roots in \mathcal{F}' (and in t' , by Remark 5.3), and let $\rho'_1 = \rho'_{r'_1 \dots r'_k} \in \Pi'$ be the element of the filled Taylor expansion $\mathcal{T}^{\mathbf{x}}(R')$ of R' associated with (t', h') and r'_1, \dots, r'_k (*i.e.* ρ'_1 is the emptying on r'_1, \dots, r'_k of the element of $\mathcal{T}(R')$ obtained via the thick subforest (t', h') of \mathcal{F}' , see Definition 5.8). By Remark 5.9, we can identify the conclusions of R' and of $\rho'_{r'_1 \dots r'_k}$. The case $a = \mathbf{exc}_i$ is trivial since it just swaps the order of two consecutive conclusions. Other cases for a :

- If $a \in \{\mathbf{cut}^i, \otimes_i, \wp_i, \wp_{ij}, \wp_{ij}, \wp_{wi}, \mathbf{1}_i, \perp_i, \mathbf{ax}_i\}$ then $\mathcal{F} = \mathcal{F}'$; let $t = t'$, $h = h'$. We verify that ρ is the element of $\mathcal{T}^{\mathbf{x}}(R)$ associated with r'_1, \dots, r'_k and the thick subforest (t, h) of \mathcal{F} .
- If $a = \mathbf{mix}_i$, let s'_1, s'_2 be the roots in \mathcal{F}' (and in t' , by Remark 5.3) corresponding to the conclusions $i, i+1$ in R' , respectively (*i.e.* $\mathbf{box}'_F(i)$ is the output of s'_1 , $\mathbf{box}'_F(i+1)$ is the output of s'_2). Let s be the root of \mathcal{F} corresponding to $i, i+1$ (*i.e.* $\mathbf{box}_F(i) = \mathbf{box}_F(i+1)$ is the output of s). We define the thick subforest (t, h) of \mathcal{F} : t is the forest obtained from t' by merging its roots s'_1 and s'_2 into the root s , and $h: t \rightarrow \mathcal{F}$ is such that $h_V(s) = s$ and $h_V(v) = h'_V(v)$ for all $v \in V_t \cap V_{t'}$. We verify that ρ is the element of $\mathcal{T}^{\mathbf{x}}(R)$ associated with r'_1, \dots, r'_k and the thick subforest (t, h) of \mathcal{F} .
- If $a = \mathbf{box}_i$, we describe the rule for non-empty box (Figure 9j); the easier rules for daimoned and empty boxes (Figures 9h and 9i) are left to the reader. Let $(t'_1, h'_1), \dots, (t'_m, h'_m)$ be thick subforests of \mathcal{F}' , let r'_1, \dots, r'_k be some roots in \mathcal{F}' (and in t'_1, \dots, t'_m , by Remark 5.3). For all $1 \leq j \leq m$, let ρ'_j be the element of $\mathcal{T}^{\mathbf{x}}(R')$ associated with (t'_j, h'_j) and r'_1, \dots, r'_k (*i.e.* ρ'_j is the emptying on r'_1, \dots, r'_k of the element of $\mathcal{T}(R')$ obtained via the thick subforest (t'_j, h'_j) of \mathcal{F}' , see Definition 5.8). The forests t'_j 's differ from each other only in the rooted tree corresponding to the conclusion i of the ρ'_j 's. Let t be the forest made of all the rooted trees on which the forests t'_1, \dots, t'_m do not differ, and of the union of the rooted trees on which the forests differ, joined by a new root; define h from h'_1, \dots, h'_m accordingly. We verify that ρ is the element of $\mathcal{T}^{\mathbf{x}}(R)$ associated with r'_1, \dots, r'_k and the thick subforest (t, h) of \mathcal{F} . \square

The rewriting system, the filled Taylor expansion (which generalizes the usual Taylor expansion) and the notion of quasi-proof-structure (which generalizes usual proof-structures) are designed to obtain the naturality result (Theorem 7.5), as shown in the example below.

Example 7.6 (the need for quasi). Extending proof-structures to quasi-proof-structures, with dashed boxes at the root level to separate their components, is crucial to achieve naturality (Theorem 7.5) when dealing with a box in a MELL proof-structure R copied zero times in an element of its Taylor expansion. Take for instance ρ_1, ρ_2 and R below.



We have $\{\rho_1, \rho_2\} \subseteq \mathcal{T}(R)$. If the action of a scheduling a were a binary relation \rightsquigarrow on MELL proof-structures and not on MELL quasi-proof-structure—ignoring the dashed boxes at the

root level in Figure 8 to separate their components—we would have $R \overset{\text{box}_2}{\rightsquigarrow} R'$. To make square (7.3) commute (as required by naturality), we observe the following facts:

- (1) the action of box_2 on ρ_2 cannot simply erase the $!$ -cell, otherwise we would lose the match with the type of R' ; so, the action of box_2 on ρ_2 must create a \boxtimes -cell of type $\mathbf{1}$;
- (2) thus, when $\{\rho_1, \rho_2\} \overset{\text{box}_2}{\rightsquigarrow} \Pi'$, the relation between Π' and R' is not $\Pi' \subseteq \mathcal{T}(R')$, because of the \boxtimes -cell; this is why we introduced the filled Taylor expansion, so that $\Pi' \subseteq \mathcal{T}^{\boxtimes}(R')$;
- (3) the two $\mathbf{1}$ -cells in R' must be handled in two different ways; indeed, the second one comes from the content of the box in R , and can be erased by the Taylor expansion of R by taking 0 copies of the box (as done in ρ_2), while the first $\mathbf{1}$ -cell cannot; to mark this difference, and to make explicit the scope of the \boxtimes -cell introduced by the action of box_2 on ρ_2 , we split the two conclusions of R into two distinct components via $R \overset{\text{mix}_1}{\rightsquigarrow} R_0$.

According to our rewrite rules, $R \overset{\text{mix}_1}{\rightsquigarrow} R_0 \overset{\text{box}_2}{\rightsquigarrow} R'_0$ and this rewriting commutes with the filled Taylor expansion on quasi-proof-structures, as required by square (7.3) for naturality.

8. GLUABILITY OF DiLL_0 QUASI-PROOF-STRUCTURES AND INHABITATION

The properties of actions of schedulings allow us to prove two of our main original results:

- (1) a characterization of the sets of DiLL_0 proof-structures that are in the Taylor expansion of some MELL proof-structure (*inverse Taylor expansion problem*, Theorem 8.3);
- (2) a characterization of the lists of MELL formulas inhabited by some cut-free MELL proof-structure (*type inhabitation problem* for cut-free MELL proof-structures, Theorem 8.10).

Both characterizations are *constructive*: when we claim the existence of an object (a MELL proof-structure with a certain property), we can actually construct a witness of it, through our rewrite rules. Thus, our characterizations can be seen as *nondeterministic* procedures, because the rewritings to build witnesses are not unique. Our procedures also *decide* the problems above in special cases of interest (Theorem 9.9, Proposition 8.12).

8.1. Gluability. We introduce the *gluability criterion* to solve the inverse Taylor expansion problem. We actually solve it in two different domains. We look for a solution:

- (1) in the set of MELL proof-structures (Theorem 8.3.1);
- (2) in the set of cut-free MELL proof-structures (Theorem 8.3.2).

We say that a scheduling is *cut-free* if does not contain the elementary scheduling cut^i .

Lemma 8.1 (cut-free scheduling). *Let R be a MELL quasi-proof-structure and ν be a scheduling such that $R \overset{\nu}{\rightsquigarrow} \vec{\varepsilon}$. We have that R is cut-free if and only if ν is cut-free.*

Proof. Let R be MELL quasi-proof-structure. If R is cut-free and $R \overset{\nu}{\rightsquigarrow} R'$, then $\nu \neq \text{cut}^i$ because the action $R \overset{\text{cut}^i}{\rightsquigarrow} R'$ requires the presence of a cut -cell in R ; moreover, R' is cut-free because no rewrite rule in Figure 8—when read left to right—introduces a cut -cell.

Conversely, if R is with cuts and $R \overset{\nu}{\rightsquigarrow} R'$ where R' is cut-free, then $\nu = \text{cut}^i$, because among the rewrite rules in Figure 8 only $\overset{\text{cut}^i}{\rightsquigarrow}$ erases a cut -cell.

A straightforward induction on the length of the scheduling ν allows us to conclude. \square

Definition 8.2 (gluability, cut-free gluability). Let Π be a set of DiLL_0 quasi-proof-structures of type Γ . If $\Pi = \emptyset$, then Π is *gluable* and *cut-free gluable*. If $\Pi \neq \emptyset$, Π is *gluable* (resp. *cut-free gluable*) if $\Pi \overset{\nu}{\rightsquigarrow} \{\vec{\varepsilon}\}$ for some scheduling (resp. cut-free scheduling) $\nu: \Gamma \rightarrow \vec{\varepsilon}$.

Theorem 8.3 (gluability criterion). *Let Π be a set of DiLL_0 proof-structures without \boxtimes -cells.*

- (1) Π is gluable if and only if $\Pi \subseteq \mathcal{T}(R)$ for some MELL proof-structure R .
- (2) Π is cut-free gluable if and only if $\Pi \subseteq \mathcal{T}(R)$ for some cut-free MELL proof-structure R .

Proof. If $\Pi = \emptyset$, Points 1–2 trivially hold. So, we can now assume $\Pi \neq \emptyset$.

- (1) If $\Pi \subseteq \mathcal{T}(R)$ for some MELL proof-structure R , then by normalization (Proposition 6.8) $R \rightsquigarrow \vec{\varepsilon}$ for some scheduling $\nu: (\Gamma) \rightarrow \vec{\varepsilon}$, where Γ is the type of R (and of each element of Π , by Remark 5.6). Therefore, $\Pi \rightsquigarrow \{\vec{\varepsilon}\}$ by naturality (Theorem 7.5), since $\emptyset \neq \Pi \rightsquigarrow \Pi'$ implies $\Pi' \neq \emptyset$ (Footnote 24), and $\mathcal{T}^{\boxtimes}(\vec{\varepsilon}) = \{\vec{\varepsilon}\}$ (Example 5.10).

Conversely, if $\Pi \rightsquigarrow \{\vec{\varepsilon}\}$ for some scheduling $\nu: (\Gamma) \rightarrow \vec{\varepsilon}$ (where Γ is the type of each element of Π), then by naturality (Theorem 7.5, as $\mathcal{T}^{\boxtimes}(\vec{\varepsilon}) = \{\vec{\varepsilon}\}$) $\Pi \subseteq \mathcal{T}^{\boxtimes}(R)$ for some MELL quasi-proof-structure $R \rightsquigarrow \vec{\varepsilon}$, and so $\Pi \subseteq \mathcal{T}(R)$ since each element of Π is without \boxtimes -cells. The (MELL) quasi-proof-structure R is indeed a proof-structure as each element of Π is a (DiLL_0) proof-structure of type Γ , and by Remark 5.6 the type of R is Γ .

- (2) Just repeat the same proof as above, paying attention that:
 - in the “if” part, if R is cut-free, then the scheduling ν is cut-free by Lemma 8.1;
 - in the “only if” part, if $\Pi \rightsquigarrow \{\vec{\varepsilon}\}$ for some cut-free scheduling $\nu: (\Gamma) \rightarrow \vec{\varepsilon}$, then by Lemma 8.1 R is cut-free, since $R \rightsquigarrow \vec{\varepsilon}$ according to naturality (Theorem 7.5). \square

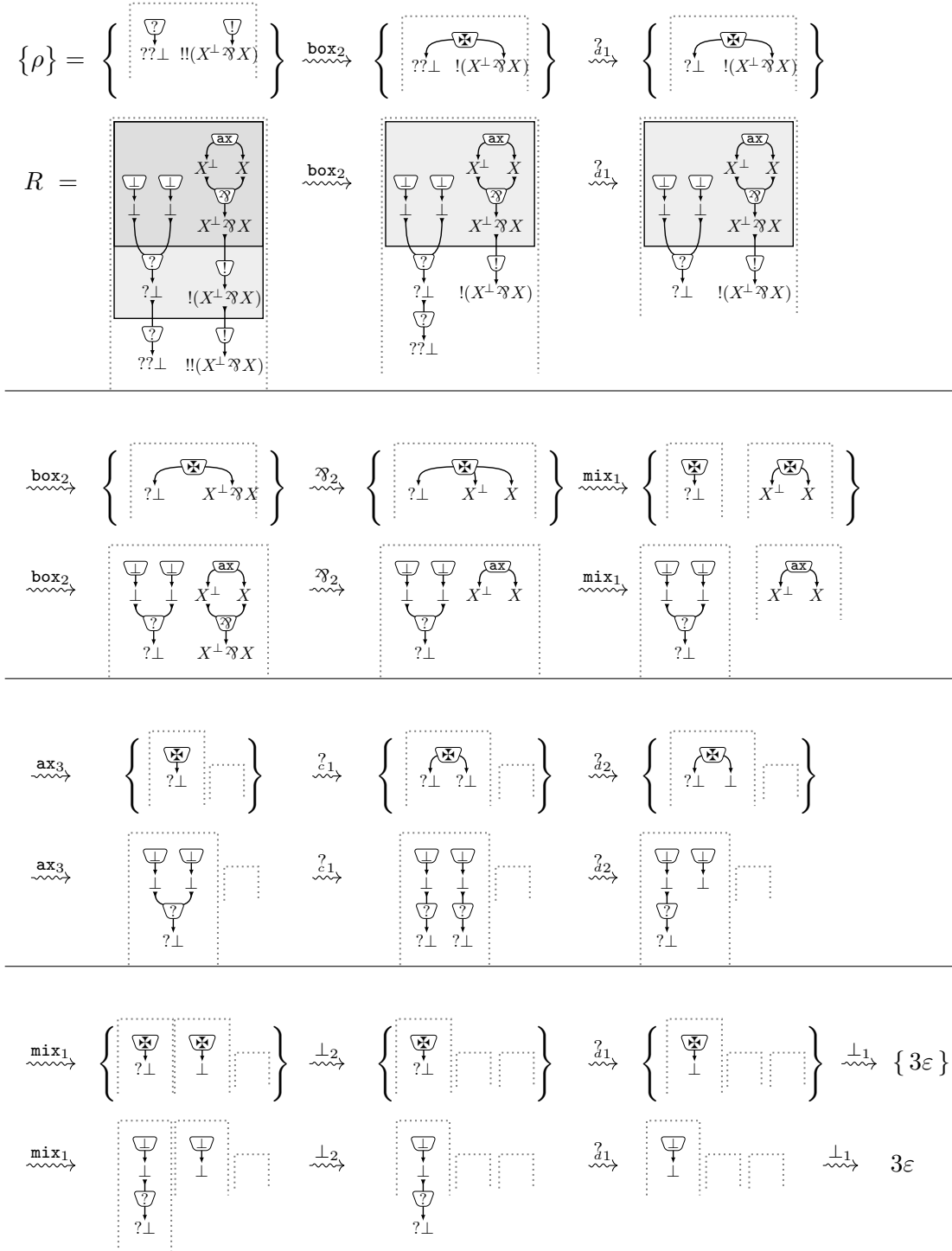
Theorem 8.3 means that the action of a scheduling rewriting a set Π of DiLL_0 proof-structures to the empty proof-structure can be seen, read in reverse, as a sequence of elementary steps to build a MELL proof structure that includes Π in its Taylor expansion.

Example 8.4. An example of the action of a scheduling starting from the singleton of a DiLL_0 proof-structure ρ and ending in $\{\vec{\varepsilon}\}$ is in Figure 11 (it is by no means the shortest possible rewriting). When replayed backward, by naturality (Theorem 7.5) it induces a MELL proof-structure R such that $\rho \in \mathcal{T}(R)$. In Figure 11 we represent simultaneously the rewriting from Π to $\{\vec{\varepsilon}\}$ and the rewriting from R to $\vec{\varepsilon}$.

We introduced two different gluability criteria (Theorems 8.3.1 and 8.3.2) because, given a non-empty set Π of *cut-free* DiLL_0 proof-structures without \boxtimes -cells, the existence of a solution to the inverse Taylor expansion problem depends on whether we look for it among MELL proof-structures or among cut-free MELL proof-structures (Example 8.5).

Example 8.5. Let $\Sigma = \{\sigma_0\}$ as in Figure 12. This singleton of a cut-free DiLL_0 proof-structure without \boxtimes -cells is gluable (as shown by the action of the scheduling in Figure 12) but not cut-free gluable (as $\{\sigma_0\} \rightsquigarrow^{\text{box}!} \{\sigma\}$ and, apart from cut^i , no elementary schedulings apply to $\{\sigma\}$ because of its type). Indeed, σ_0 “masks” any information about the content B of the box represented by its $!$ -cell, apart from its type $!X$ which does not allow B to be a cut-free MELL proof-structure (see also Section 8.2). Gluability (with cuts) of $\{\sigma\}$ is related to the fact that there is a MELL proof-structure of type X with cuts, for every atomic formula X (see Remark 8.8). Note that $\sigma_0 \in \mathcal{T}(S_0)$, and $\sigma \in \mathcal{T}^{\boxtimes}(S)$ and $\sigma' \in \mathcal{T}^{\boxtimes}(S')$, where S_0 , S and S' are the MELL proof-structures with cuts represented in Figure 12.

Example 8.6. The three DiLL_0 proof-structures ρ_1, ρ_2, ρ_3 below are not gluable as a whole, but are gluable two by two (this is a slight variant of the example in [Tas09, pp. 244–246]). In fact, there is no MELL proof-structure whose Taylor expansion contains ρ_1, ρ_2, ρ_3 , but any pair of them is in the Taylor expansion of some MELL proof-structure.



Scheduling $\text{box}_2 ?d_1 \text{box}_2 \wp_2 \text{mix}_1 \text{ax}_3 ?e_1 ?d_2 \text{mix}_1 \perp_2 ?d_1 \perp_1 : (??\perp, !!(X^\perp \wp X)) \rightarrow 3\epsilon$

FIGURE 11. Action of a scheduling witnessing that $\rho \in \mathcal{T}(R)$.

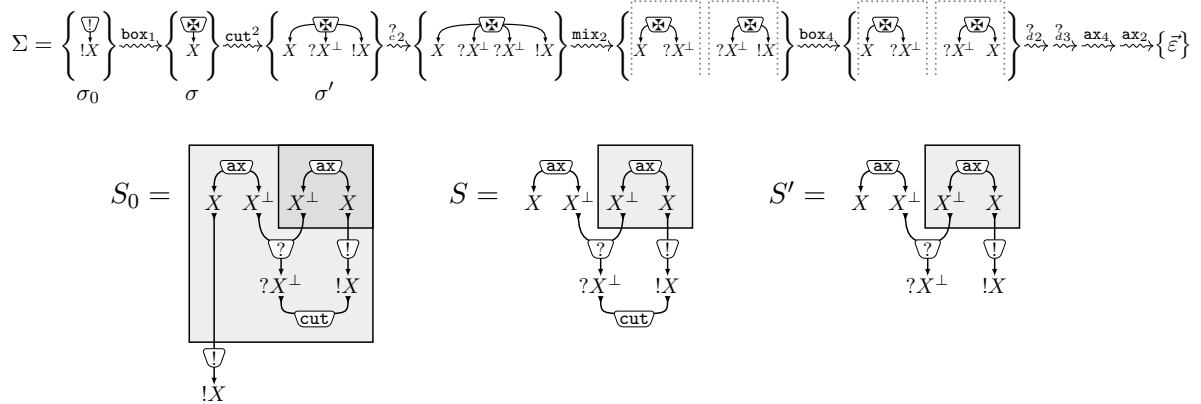
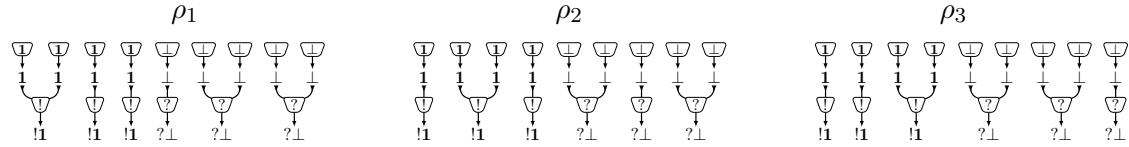
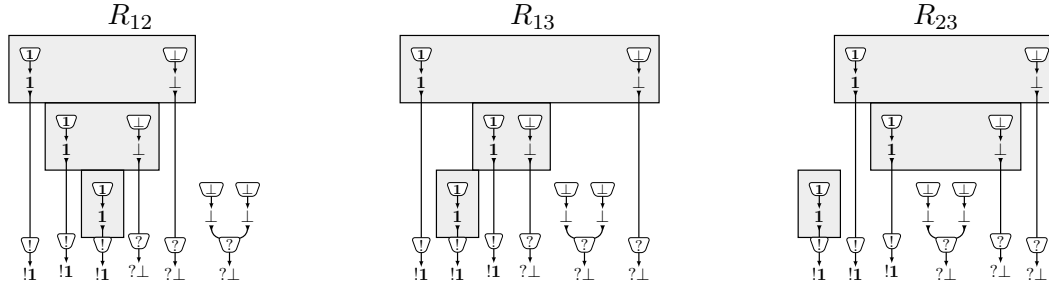


FIGURE 12. Example of gluability that is not cut-free gluability.



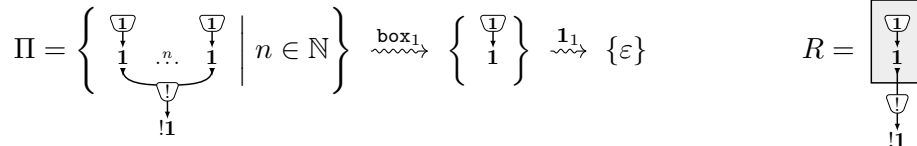
Indeed, for any $i, j \in \{1, 2, 3\}$ with $i \neq j$, ρ_i and ρ_j are in the Taylor expansion of R_{ij} below.



Example 8.6 can be generalized to finite sets of any cardinality: for any $n > 1$, there is a finite set Π of $n + 1$ DiLL_0 proof-structures that is not gluable as a whole, but all subsets of Π of at most n elements are gluable. This means that, for every $n > 1$, gluability of a set Π of DiLL_0 proof-structures is not reducible to test that all subsets of Π with at most n elements satisfy some n -ary coherence relation, unless (obviously) Π is a finite set with at most n elements. As explained in Section 1, this difficulty in the inverse Taylor expansion problem is typical of MELL and does not arise in the λ -calculus.

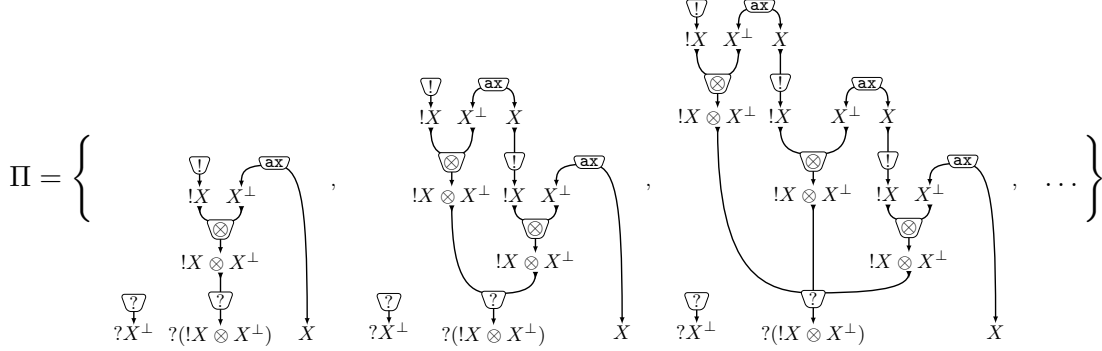
Remark 8.7 (infinite sets). The gluability criterion (Theorem 8.3) is not limited to finite sets of DiLL_0 proof-structures, it holds for infinite sets too. This raises the question of the relationship between gluability and finite gluability, namely: when every finite subset of a given infinite set of DiLL_0 proof-structures is gluable, is the infinite set itself gluable too? The answer is negative. Let Π be an *infinite* set of DiLL_0 proof-structures such that *every finite* subset of Π is gluable. There are two cases for Π .

- Π itself is gluable. It is typically the case when Π is exactly the Taylor expansion of a MELL proof-structure R with at least one box, such as in the following example.



We might say that Π is *infinite in width*: infinity is related to the number of copies chosen for the boxes of R , a number that has to be finite but can be arbitrarily large.

- Π is not gluable. This is the case in the following set (we draw three elements only).



Every finite subset of Π is gluable, but the whole infinite set Π is not. We might say that Π is *infinite in depth*: in Π , for every $n > 0$ there is a DiLL_0 proof-structure with n $!$ -cells. Any finite subset Π_n of Π (n is the maximal number of $!$ -cells in the elements of Π_n) is included in the Taylor expansion of a MELL proof-structure with depth n , and indeed the action of the scheduling ν_n^1 defined in Example 7.3 rewrites Π_n to $\{\bar{\varepsilon}\}$.²⁶ However, the whole Π can only be seen as the Taylor expansion of an “infinite” proof-structure with infinitely deep nested boxes. Such infinite proof-structures are not in the syntax of MELL. An ongoing research relates them to non-well-founded proofs for fixed-point logics [DS19, DPS21].

In the inverse Taylor expansion problem, both kinds of infinity—in width and in depth—arise already in the λ -calculus, they are not specific to MELL. An instance of a set of resource λ -terms that is infinite in width is the Taylor expansion of the λ -term xy . The set $\Pi = \{\lambda f.\lambda x.f[], \lambda f.\lambda x.f[f[]], \lambda f.\lambda x.f[f[f[]]], \dots\}$ of resource λ -terms is an example of infinity in depth: every finite subset of Π is included in the Taylor expansion of some λ -term (a Church numeral), but Π is not included in the Taylor expansion of any λ -term.

8.2. Inhabitation. We can characterize type inhabitation in MELL proof-structures. We first put the question of type inhabitation for MELL proof-structures in a proper way.

Remark 8.8 (inhabitation with cuts is trivial). For every atomic formula X there is a MELL proof-structure of type X with cuts, the one called S in Figure 12. More generally, for every MELL formula A (resp. list Γ of MELL formulas), there is a MELL proof-structure of type A (resp. Γ) possibly with cuts. Indeed, for A just “ η -expand” S in Figure 12, by replacing each ax -cell in S with the “canonical” MELL proof-structure of type A, A^\perp . For $\Gamma = (A_1, \dots, A_n)$, repeat the same procedure for each A_i separately, and juxtapose the MELL proof-structures to yield a unique MELL proof-structure R of type Γ (take $R = \varepsilon$ if $\Gamma = \varepsilon$).

Therefore, type inhabitation for MELL proof-structures is trivial, if cut-cells are allowed.

Type inhabitation for MELL proof-structures is a non-trivial problem only if we restrict to *cut-free* MELL proof-structures. Indeed, given two propositional variables $X \neq Y, X$, there is no *cut-free* MELL proof-structure of type X or Y, X . So, the problem is the following:

Data: a list Γ of MELL formulas;

Question: does there exist a cut-free MELL proof-structure of type Γ ?

²⁶In fact, if π_h is the element of such a Π with exactly h $!$ -cells, $\{\pi_h\} \xrightarrow{\nu_n^1} \{\bar{\varepsilon}\}$ for all $n \geq h$.

We study the *type inhabitation problem for cut-free MELL proof-structures* via schedulings, following the idea that a scheduling $\nu: (\Gamma) \rightarrow \vec{\epsilon}$ “encodes” a MELL proof-structure of type Γ .

Let us see the behavior of *cut-free* schedulings. For some list Γ of MELL formulas there is no cut-free scheduling $\nu: (\Gamma) \rightarrow \vec{\epsilon}$. For instance, if $X \neq Y$ are propositional variables and we exclude cut^i , then there are no elementary schedulings from (X) , while the only other elementary schedulings from (Y, X) are mix_1 and exc_1 . The non-existence of a cut-free scheduling from (X) or (Y, X) to $\vec{\epsilon}$ goes hand in hand with the non-existence of a cut-free MELL proof-structure of type X or Y, X , by Proposition 6.8 and Lemma 8.1.

Conversely, for any list Γ of MELL formulas, a MELL proof-structure $R(\nu)$ of type Γ is associated with every scheduling $\nu: (\Gamma) \rightarrow \vec{\epsilon}$: such a $R(\nu)$ is built by reading ν backward, it exists and is uniquely determined because the action of ν is *co-functional* (Lemma 6.5).

Definition 8.9 (building from scheduling). Let $\Gamma \neq \epsilon$ be a list of lists of MELL formulas, and $\nu: \Gamma \rightarrow n\epsilon$ be a scheduling with $n > 0$. The MELL quasi-proof-structure $R(\nu)$ is defined by induction on the length of ν : if ν is empty, $R(\nu) = n\epsilon$; otherwise $\nu = a\mu$ for some elementary scheduling a , and $R(\nu)$ is the only MELL quasi-proof-structure R such that $R \xrightarrow{\nu} R(\mu)$.

Clearly, for any scheduling $\nu: \Gamma \rightarrow \vec{\epsilon}$, $R(\nu)$ is of type Γ and $R(\nu) \xrightarrow{\nu} \vec{\epsilon}$, and $R(\nu)$ is cut-free if and only if so is ν (Lemma 8.1). Actually, cut-free schedulings are a way to characterize the (lists of) MELL formulas that are the type of some cut-free MELL proof-structure.

Theorem 8.10 (type inhabitation for cut-free MELL proof-structures). *Let Γ be a list of MELL formulas. There is a cut-free MELL proof-structure of type Γ if and only if there is a cut-free scheduling $\nu: (\Gamma) \rightarrow \vec{\epsilon}$.*

Proof. If $\nu: (\Gamma) \rightarrow \vec{\epsilon}$ is a cut-free scheduling, $R = R(\nu)$ is a MELL quasi-proof-structure of type Γ . As Γ is a list of MELL formulas, R is a proof-structure. By Lemma 8.1, R is cut-free.

Conversely, if R is a cut-free MELL proof-structure of type Γ , then $R \xrightarrow{\nu} \vec{\epsilon}$ for some cut-free scheduling $\nu: (\Gamma) \rightarrow \vec{\epsilon}$, by Proposition 6.8 and Lemma 8.1. \square

Lemma 6.5, Definition 8.9 and the proof of Theorem 8.10 say that, given a list Γ of MELL formulas, any cut-free scheduling from (Γ) to $\vec{\epsilon}$ can be seen, when read backward, as a sequence of elementary steps to build a cut-free MELL proof-structure of type Γ from scratch.

Consider now the *procedure* \mathcal{P}_1 below, given a list Γ of MELL formulas as input:

- (1) for all $n \geq 0$, take all cut-free schedulings from (Γ) of length n (there are finitely many);
- (2) as soon as a list of the form $\vec{\epsilon}$ is reached by a cut-free scheduling, accept Γ .

Theorem 8.10 guarantees that such a *nondeterministic* procedure accepts Γ if and only if there is a cut-free MELL proof-structure of type Γ , and any cut-free scheduling ν that allows the procedure to accept Γ also constructs a witness $R(\nu)$. Therefore, we can conclude:

Fact 8.11 (semi-decidability). Procedure \mathcal{P}_1 semi-decides the type inhabitation problem for cut-free MELL proof-structures.

Procedure \mathcal{P}_1 need not halt. Indeed, in general, there is no bound on the length of cut-free schedulings, and the elementary scheduling $?_i$ can create cut-free schedulings of arbitrary length with target $\vec{\epsilon}$, such as μ_n below—of length $3n + 1$ —for every $n \geq 0$.

$$\mu_n = \underbrace{?_1 \cdots ?_1}_{n \text{ times}} \underbrace{\text{mix}_1 ?_1 \cdots ?_1}_{n \text{ times}} : (?X) \rightarrow (n + 1)\epsilon$$

Consider now the *procedure* \mathcal{P}_2 below, a slight variant of \mathcal{P}_1 for the same input Γ :

- (1) for all $n \geq 0$, take all cut-free schedulings from (Γ) of length n that do not contain the elementary scheduling $?_i$ (there are finitely many);
- (2) as soon as a list of the form \vec{e} is reached by one of those schedulings, accept Γ .

Procedure \mathcal{P}_2 always halts. Indeed, define the following *sizes*:

- the size $\mathfrak{s}(A)$ of a MELL formula A is the number of atoms, connectives and units occurring in A , *i.e.* $\mathfrak{s}(X) = \mathfrak{s}(\mathbf{1}) = 1$, $\mathfrak{s}(A \otimes B) = \mathfrak{s}(A) + \mathfrak{s}(B) + 1$, $\mathfrak{s}(!A) = \mathfrak{s}(A) + 1$ and $\mathfrak{s}(A^\perp) = \mathfrak{s}(A)$;
- the size of a list (A_1, \dots, A_n) of MELL formulas is the sum $\sum_{i=1}^n \mathfrak{s}(A_i)$;
- the size of a (non-empty) list Γ of lists of MELL formulas is $\mathfrak{s}(\Gamma) = (|\Gamma|, \mathfrak{n}(\Gamma))$ with the lexicographic order, where $|\Gamma|$ is the size of the flattening of Γ , and $\mathfrak{n}(\Gamma)$ is the finite multiset of the lengths of each list in Γ , with the multiset order.

One has $\mathfrak{s}(\Gamma) > \mathfrak{s}(\Gamma')$ for any elementary scheduling $a: \Gamma \rightarrow \Gamma'$ except

- when $a = \text{cut}^i$ since $\mathfrak{s}(\Gamma) < \mathfrak{s}(\Gamma')$, but \mathcal{P}_2 considers cut-free schedulings only;
- when $a = \text{exc}_i$ since $\mathfrak{s}(\Gamma) = \mathfrak{s}(\Gamma')$, but infinite chains of consecutive exc_i are irrelevant, because for every Γ , sufficiently long schedulings $\text{exc}_{i_1} \cdots \text{exc}_{i_n}$ are from Γ to Γ ;²⁷
- when $a = ?_i$ since $\mathfrak{s}(\Gamma) < \mathfrak{s}(\Gamma')$, but \mathcal{P}_2 only considers schedulings without $?_i$.

The nondeterministic procedure \mathcal{P}_2 decides the type inhabitation problem in a subset of MELL proof-structures. We say that a $?_i$ -cell with a least two inputs is a *strict contraction*. Note that if $a = ?_i$ and $R \xrightarrow{a} R'$ is not nullary $?_i$ -splitting, then R has a strict contraction.

Proposition 8.12 (decidability without strict contractions). *Procedure \mathcal{P}_2 decides the type inhabitation problem for cut-free MELL proof-structures without strict contractions, that is,*

Data: a list Γ of MELL formulas;

Question: *is there a cut-free MELL proof-structure of type Γ without strict contractions?*

Proof. Let Γ be a list of MELL formulas. Termination of \mathcal{P}_2 on input Γ is proved above.

If \mathcal{P}_2 accepts Γ , then there is a cut-free scheduling $\nu: (\Gamma) \rightarrow \vec{e}$ that does not contain $?_i$. According to Definition 8.9, it is immediate to prove that $R(\nu)$ has no strict contractions.

If there is a cut-free MELL proof-structure R of type (Γ) without strict contractions, then there is a scheduling $\nu: (\Gamma) \rightarrow \vec{e}$ such that $R \xrightarrow{\nu} \vec{e}$ without nullary $?_i$ -splittings (Proposition 6.8). Since R has no strict contractions, then ν does not contain any $?_i$, so \mathcal{P}_2 accepts Γ . \square

Type inhabitation for cut-free MELL proof-structures depends on whether strict contractions are allowed or not: the list $!X, ?X^\perp, !X$ is accepted by \mathcal{P}_1 but rejected by \mathcal{P}_2 , indeed the only cut-free MELL proof-structure of type $!X, ?X^\perp, !X$ has a strict contraction.

Remark 8.13 (on decidability of MELL). Our characterization of type inhabitation for cut-free MELL proof-structures (Theorem 8.10) can be seen in relation to the question of deciding if a MELL formula is provable: indeed, a MELL formula A is provable if and only if there is a *correct* cut-free MELL proof-structure of type A (for some notion of correctness).

In the propositional setting, MLL provability is decidable and **NP**-complete [Kan94], while LL provability is undecidable [LMSS92]; a longstanding open problem is whether provability in MELL is decidable or not. Our result offers a fresh perspective on it.

Take again procedure \mathcal{P}_1 above with the same input Γ , and replace its second step by:

- (2) if a list of the form \vec{e} is reached by a cut-free scheduling ν , construct the cut-free MELL proof-structure $R = R(\nu)$ of type Γ ;

²⁷Indeed, for every list $\Gamma \neq \epsilon$ of lists of MELL formulas, there is $N_\Gamma \geq 0$ such that, for every scheduling $\nu: \Gamma \rightarrow \Delta$ only made of exc_i , there is a scheduling $\mu: \Gamma \rightarrow \Delta$ only made of exc_i and of length $\leq N_\Gamma$.

- (3) check if R fulfills a suitable correctness criterion that characterizes all and only the MELL proof-structures corresponding to a derivation in MELL sequent calculus, *e.g.* [TdF03];
- (4) as soon as a correct (cut-free) MELL proof-structure is found, accept Γ .

As explained above, there may be infinitely many cut-free schedulings $\nu: (\Gamma) \rightarrow \vec{\epsilon}$, and also infinitely many cut-free MELL proof-structures of type Γ . But they are finite in number in the case without strict contractions. As correctness of a MELL proof-structure is decidable, the procedure above *semi-decides* provability in MELL (is a given sequent provable in MELL sequent calculus?) and *decides* provability in MELL *without contractions* (is a given sequent provable in MELL sequent calculus without the contraction rule?). Our procedure restricts its search to cut-free objects, without loss of generality by cut-elimination. Semi-decidability of MELL provability and decidability of MELL provability without contractions are not new results; another (and simpler) proof relies on the fact that in MELL sequent calculus every inference rule other than contraction and cut, read bottom-up, decreases the size of a sequent, for a suitable notion of size. What is new is the analogy between the type inhabitation problem for cut-free MELL proof-structures and the provability problem in MELL.

Summing up, (strict) contraction is the only reason that makes hard to prove whether type inhabitation for cut-free MELL proof-structures, as well as MELL provability, are decidable or not. This seems to suggest that *correctness* (the fact that a MELL proof-structure is a proof of a MELL formula) does not play an essential role in the question if MELL provability is decidable or not. However, for the time being, we are not able to reduce decidability of provability in MELL to decidability of type inhabitation for cut-free MELL proof-structures.

9. DECIDABILITY OF THE FINITE INVERSE TAYLOR EXPANSION PROBLEM

Consider the decision problem below, called the *finite inverse Taylor expansion problem*.

Data: a *finite* set Π of DiLL_0 proof-structures without \boxtimes -cells;

Question: does there exist a MELL proof-structure R such that $\Pi \subseteq \mathcal{T}(R)$?

By Theorem 8.3, a positive answer is equivalent to the gluability of Π , thus the question amounts to decide the gluability of a finite set Π of DiLL_0 proof-structures without \boxtimes -cells. In this section, we show that the problem is actually *decidable*, using our rewrite rules. The challenge here is to restrict the search space for rewritings to guarantee termination.

Recall that the action of box_i is the only rewrite rule in DiLL_0 that can introduce \boxtimes -cells (Figure 9i). We identify \boxtimes -cells that have the same number of outputs, with the same type.

Remark 9.1 (\boxtimes -cell). Let Σ, Π be sets of DiLL_0 quasi-proof-structures such that every element of Σ is without \boxtimes -cells, $\Sigma \rightsquigarrow \Pi$ and Π is gluable. The presence of \boxtimes -cells in Π witnesses the absence of information on the content of some box of some element of Σ . However, in order that Π is gluable, either this lack of information in some element of Π is provided by another element of Π , or the information is plainly missing. Indeed, let Π be a set of DiLL_0 proof-structures, and i be a conclusion of some $\rho \in \Pi$ and the output of a \boxtimes -cell of ρ ; we have two cases (Definition 9.3 below relies on this distinction):

- (1) either i is the output of a ℓ -cell of *some* DiLL_0 quasi-proof-structure $\pi \in \Pi$, with $\ell \neq \boxtimes$;
- (2) or i is the output of a \boxtimes -cell of *every* DiLL_0 quasi-proof-structure $\pi \in \Pi$; in this case the component of every $\pi \in \Pi$ having i among its conclusions consists of the same \boxtimes -cell.

Let $n > 0$ and $\Gamma = (\Gamma_1; \dots; \Gamma_n)$ be a list of lists of MELL formulas. We denote by \boxtimes_Γ the DiLL_0 quasi-proof-structure $(\rho_{\Gamma_1}, \dots, \rho_{\Gamma_n})$ of type Γ , where ρ_{Γ_i} is the daimon of type Γ_i

if $\Gamma_i \neq \epsilon$, and $\rho_{\Gamma_i} = \epsilon$ if $\Gamma_i = \epsilon$. Lemma 9.2 below states that every $\{\mathfrak{X}_\Gamma\}$ is gluable: an example for $\Gamma = (X)$ with X atomic is in Figure 12 (see the daimon σ there). A set Π of DiLL_0 quasi-proof-structures is \mathfrak{X} -full if $\Pi = \{\mathfrak{X}_\Gamma\}$ for some Γ . Note that any $\{\bar{\epsilon}\}$ is \mathfrak{X} -full.

An elementary scheduling cut^i is *with conclusion* if $\Gamma_k \neq \epsilon$ in its definition in Figure 7.

Lemma 9.2 (gluability of daimons). *Every \mathfrak{X} -full set of DiLL_0 quasi-proof-structures is gluable. More precisely, let Γ be a non-empty list of lists of MELL formulas. Then,*

- (1) *there is a scheduling $\nu: \Gamma \rightarrow \bar{\epsilon}$ such that every cut^i in ν is with conclusion;*
- (2) *if $\nu: \Gamma \rightarrow \bar{\epsilon}$ and every cut^i in ν is with conclusion, then $\{\mathfrak{X}_\Gamma\} \rightsquigarrow \{\bar{\epsilon}\}$.*

Proof. By Points 1 and 2, every \mathfrak{X} -full set of DiLL_0 quasi-proof-structures is gluable.

- (1) For every MELL formula A , we define by induction on A a scheduling $\nu_A: (A) \rightarrow \bar{\epsilon}$ where every cut^i in ν_A is with conclusion. We shall use the fact that for every MELL formula A and lists Δ, Γ of MELL formulas, given a scheduling $\nu_A: (A) \rightarrow \bar{\epsilon}$, one has $\nu_A: (A, \Delta; \Gamma) \rightarrow (\Delta; \Gamma)$ and $\nu_A: (\epsilon; A, \Gamma) \rightarrow (\epsilon; \Gamma)$. Let $\mu_A = \text{cut}_2^2 \text{mix}_2 \text{box}_4 \text{d}_2 \text{d}_3: (A) \rightarrow (A, A^\perp; A^\perp, A)$ and note that cut^2 in μ_A is with conclusion.

- If $A = X$ (and similarly if $A = X^\perp$), then $\nu_A = \mu_A \text{ax}_4 \text{ax}_2$.
- If $A = \mathbf{1}$ then $\nu_A = \mathbf{1}_1$; if $A = \perp$ then $\nu_A = \perp_1$.
- If $A = B \otimes C$ (similarly if $A = B \wp C$), $\nu_A = \mu_A \otimes_4 \wp_3 \wp_2 \otimes_1 \nu_B \nu_C \nu_{B^\perp} \nu_{C^\perp} \nu_{B^\perp} \nu_{C^\perp} \nu_{B^\perp} \nu_C$.
- If $A = !B$ (and similarly if $A = ?B$), then $\nu_A = \mu_A \text{box}_4 \text{d}_3 \text{box}_1 \text{d}_2 \nu_B \nu_{B^\perp} \nu_{B^\perp} \nu_B$.

Let $\Gamma = (A_1, \dots, A_{i_1}; \dots; A_{i_{n-1}+1}, \dots, A_m)$ be a non-empty list of lists of MELL formulas. In the scheduling $\nu_{A_1} \dots \nu_{A_{i_1}} \dots \nu_{A_{i_{n-1}+1}} \dots \nu_{A_m}: \Gamma \rightarrow \bar{\epsilon}$ every cut^i is with conclusion.

- (2) By induction on the length $k \geq 0$ of the scheduling $\nu = a_1 \dots a_k$, where a_j is an elementary scheduling for all $1 \leq j \leq k$. It is crucial that every $a_j = \text{cut}^i$ is with conclusion: otherwise, for a daimon ρ' there is no ρ such that $\{\rho\} \rightsquigarrow^{\text{cut}^i} \{\rho'\}$ (see Figure 9d). \square

Therefore, by Lemma 9.2, to prove that a set Π of DiLL_0 proof-structures is gluable, it suffices to show that Π rewrites to a \mathfrak{X} -full set Π' . A more subtle question is to prove that if Π cannot rewrite to a \mathfrak{X} -full set Π' , then Π is not gluable (Lemma 9.5 below).

Definition 9.3 (\mathfrak{X} -/ $\neg\mathfrak{X}$ -actions, canonicity). Let Π be a set of DiLL_0 quasi-proof-structures with the same conclusions. Suppose that $\Pi \rightsquigarrow \Pi'$ where $a \neq \text{cut}^i$ (resp. $a = \text{cut}^i$) is an elementary scheduling that acts on the conclusion i of all elements of Π (resp. Π'). If for every $\rho \in \Pi$ (resp. $\rho' \in \Pi'$) the conclusion i of ρ (resp. ρ') is the output of a \mathfrak{X} -cell, then we say that $\Pi \rightsquigarrow \Pi'$ is a \mathfrak{X} -action. If $\Pi \rightsquigarrow \Pi'$ and for every elementary scheduling a such that $\Pi \rightsquigarrow \Pi' = \Pi \rightsquigarrow^1 \Pi_1 \rightsquigarrow \Pi_2 \rightsquigarrow^2 \Pi'$ one has that $\Pi_1 \rightsquigarrow \Pi_2$ is (resp. is not) a \mathfrak{X} -action, we say that $\Pi \rightsquigarrow \Pi'$ is a \mathfrak{X} -action (resp. $\neg\mathfrak{X}$ -action), written $\Pi \rightsquigarrow^{\mathfrak{X}} \Pi'$ (resp. $\Pi \rightsquigarrow^{\neg\mathfrak{X}} \Pi'$).

If $\Pi \rightsquigarrow \{\bar{\epsilon}\} = \Pi \rightsquigarrow^{\mathfrak{X}} \{\bar{\epsilon}\}$, then $\Pi \rightsquigarrow \{\bar{\epsilon}\}$ is *canonical* and Π is *canonically gluable*.

So, given a set Π_0 of DiLL_0 quasi-proof-structures and a \mathfrak{X} -action $\Pi_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} \Pi_n$, for all $1 \leq j \leq n$ the elementary scheduling a_j acts on a conclusion of a daimon in *all* elements of Π_{j-1} . On the other hand, in a $\neg\mathfrak{X}$ -action $\Pi_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} \Pi_n$, for all $1 \leq j \leq n$ the elementary scheduling a_j acts on the output of a ℓ -cell with $\ell \neq \mathfrak{X}$ in *some* element of Π_{j-1} .

Remark 9.4 (fullness). Let Π be a set of DiLL_0 quasi-proof-structures, and ν be a scheduling. By straightforward induction on the length of ν , we can prove the facts below.

- If Π is \mathfrak{X} -full and $\Pi \rightsquigarrow \Pi'$ for some set Π' , then $\Pi \rightsquigarrow \Pi'$ is a \mathfrak{X} -action and Π' is \mathfrak{X} -full.

- If $\Pi \rightsquigarrow \{\varepsilon\}$, then $\Pi \rightsquigarrow \{\varepsilon\}$ is a \blacktriangleright -action if and only if Π is \blacktriangleright -full. Therefore, $\Pi \rightsquigarrow \{\varepsilon\}$ is canonical if and only if $\Pi \rightsquigarrow \{\varepsilon\} = \Pi \rightsquigarrow \Pi' \rightsquigarrow \{\varepsilon\}$ for some \blacktriangleright -full Π' .

The following lemma relies on the fact that \blacktriangleright -actions can always be delayed.

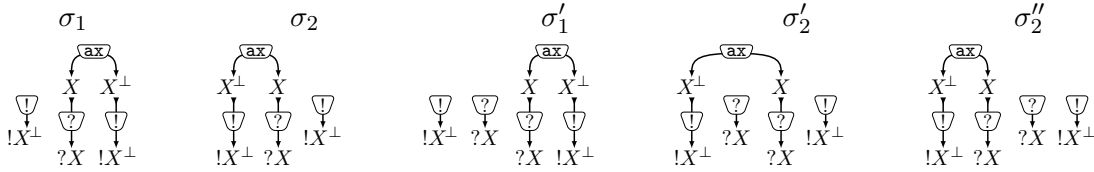
Lemma 9.5 (postponement of \blacktriangleright -actions). *Every gluable set of DiLL_0 quasi-proof-structures is canonically gluable.*

Proof. Immediate consequence of the following fact: if Π_1, Π_2, Π_3 are sets of DiLL_0 quasi-proof-structures such that $\Pi_1 \xrightarrow{a_1} \Pi_2$ is a \blacktriangleright -action and $\Pi_2 \xrightarrow{a_2} \Pi_3$ is not a \blacktriangleright -action for some elementary schedulings a_1, a_2 , then there are elementary schedulings c_1, c_2 and a set Σ of DiLL_0 quasi-proof-structures such that $\Pi_1 \xrightarrow{c_1} \Sigma$ is not a \blacktriangleright -action and $\Sigma \xrightarrow{c_2} \Pi_3$ is a \blacktriangleright -action. Notice that c_1 (resp. c_2) is “essentially” the same as a_2 (resp. a_1). \square

We have defined nullary $?$ -splitting actions on MELL quasi-proof-structures in Section 6. This notion can be easily extended to DiLL_0 . Let Π be a set of DiLL_0 quasi-proof-structures. Given an elementary scheduling a , $\Pi \rightsquigarrow \Pi'$ is *nullary $?$ -splitting* if $a = ?_i$ and, for every $\rho \in \Pi$, it splits the $?$ -cell of ρ with output i and $h_\rho \geq 0$ inputs into two $?$ -cells, one $?$ -cell with output i (resp. $i + 1$) and h_ρ inputs, the other $?$ -cell with output $i + 1$ (resp. i) and 0 inputs. Given a scheduling ν such that $\Pi \rightsquigarrow \Pi'$, $?_c^0(\Pi \rightsquigarrow \Pi')$ is the number of nullary $?$ -splitting actions in $\Pi \rightsquigarrow \Pi'$, that is, the number of elementary schedulings $a = ?_i$ such that $\Pi \rightsquigarrow \Pi' = \Pi \rightsquigarrow \Pi_1 \rightsquigarrow \Pi_2 \rightsquigarrow \Pi'$ and $\Pi_1 \rightsquigarrow \Pi_2$ is nullary $?$ -splitting. We say that:

- $\Pi \rightsquigarrow \Pi'$ is *without nullary $?$ -splittings* if $?_c^0(\Pi \rightsquigarrow \Pi') = 0$; *has nullary $?$ -splittings* otherwise;
- Π is *canonically gluable without nullary $?$ -splittings* if there is a scheduling ν such that $\Pi \rightsquigarrow \{\varepsilon\}$ is canonical and without nullary $?$ -splittings.

For instance, in Example 7.3, $\{\pi\} \rightsquigarrow \{\pi'\}$ is nullary $?$ -splitting. And $\{\sigma_1, \sigma_2\} \rightsquigarrow \Sigma$ is nullary $?$ -splitting if $\Sigma = \{\sigma'_1, \sigma'_2\}$, but it is not nullary $?$ -splitting if $\Sigma = \{\sigma'_1, \sigma''_2\}$, where $\sigma_1, \sigma_2, \sigma'_1, \sigma'_2, \sigma''_2$ are the DiLL_0 proof-structures below.



Remark 9.6 (need for nullary splitting). For π_h ($h > 0$) as in Footnote 26 on p. 33, one has $\{\pi_1, \pi_2\} \rightsquigarrow \{\varepsilon\}$ only with schedulings ν whose action on $\{\pi_1\}$ has nullary $?$ -splittings. So, nullary $?$ -splittings are somehow “necessary” in DiLL_0 , differently from MELL where they are superfluous for normalization (Proposition 6.8). To make diagram (7.2) commute for naturality (Theorem 7.5), nullary $?$ -splitting actions in DiLL_0 must be mimicked by nullary $?$ -splitting actions on MELL , hence the action of $?_i$ in MELL must allow nullary $?$ -splittings.

Akin to MELL , nullary $?$ -splitting actions in DiLL_0 are superfluous as means of destruction. They may be needed for *some* elements in a set Π of DiLL_0 quasi-proof-structures, but not for *all* of them, thus nullary $?$ -splitting actions are useless for the set Π as a whole.

Lemma 9.7 (splitting in DiLL_0). *Every canonically gluable set Π of DiLL_0 quasi-proof-structures is canonically gluable without nullary $?$ -splittings.*

Proof. By Remark 9.4, it suffices to prove that for every canonically gluable set Π of DiLL_0 quasi-proof-structures, there is a scheduling ν such that $\Pi \rightsquigarrow \Pi'$ is a $\rightarrow\blacktriangleright$ -action without

$?_c$ -splittings, for some \boxtimes -full Π' . Indeed, by Lemma 9.2, there is a scheduling μ such that $\Pi' \xrightarrow{\mu} \{\bar{\varepsilon}\}$, it is a \boxtimes -action by Remark 9.4 and is clearly without nullary $?_c$ -splittings.

Given a $\neg\boxtimes$ -action $\Pi \xrightarrow{\nu} \Pi'$ where Π' is \boxtimes -full, we prove by induction on $?_c^0(\Pi \xrightarrow{\nu} \Pi') \in \mathbb{N}$ that it is possible to build a $\neg\boxtimes$ -action $\Pi \xrightarrow{\nu'} \Pi''$ without nullary $?_c^0$ -splittings for some \boxtimes -full Π'' . Clearly, it is enough to show that if $?_c^0(\Pi \xrightarrow{\nu} \Pi') > 0$ then we can find a \boxtimes -full Π'' and a $\neg\boxtimes$ -action $\Pi \xrightarrow{\nu'} \Pi''$ such that $?_c^0(\Pi \xrightarrow{\nu} \Pi') > ?_c^0(\Pi \xrightarrow{\nu'} \Pi'')$.

Let $\Pi \xrightarrow{\nu} \Pi' = \Pi \xrightarrow{\nu_1} \Pi_1 \xrightarrow{a} \Pi_2 \xrightarrow{\nu_2} \Pi'$ be a $\neg\boxtimes$ -action, where Π' is \boxtimes -full, $a = ?_c^k$ and $\Pi_1 \xrightarrow{a} \Pi_2$ is nullary $?_c$ -splitting. Since the $?_c$ -cells with 0 inputs of Π_2 created by $\Pi_1 \xrightarrow{a} \Pi_2$ must disappear in Π' (which contains only \boxtimes -cells), there are only two cases for $\Pi_2 \xrightarrow{\nu_2} \Pi'$:

- (1) either $\Pi_2 \xrightarrow{\nu_2} \Pi' = \Pi_2 \xrightarrow{\nu_{21}} \Pi_3 \xrightarrow{w_i} \Pi_4 \xrightarrow{\nu_{22}} \Pi'$, where $\Pi_3 \xrightarrow{w_i} \Pi_4$ erases the $?_c$ -cells without inputs and their unique output (a conclusion of each element of Π_3) created by $\Pi_1 \xrightarrow{a} \Pi_2$;
- (2) or $\Pi_2 \xrightarrow{\nu_2} \Pi' = \Pi_2 \xrightarrow{\nu_{21}} \Pi_3 \xrightarrow{\text{box}_i} \Pi_4 \xrightarrow{\nu_{22}} \Pi'$, where the component of every element of Π_3 containing the $?_c$ -cell without inputs created by $\Pi_1 \xrightarrow{a} \Pi_2$ is transformed by $\Pi_3 \xrightarrow{\text{box}_i} \Pi_4$ into a \boxtimes -cell, according to the rewrite rule in Figure 9i applied to all the elements of Π_3 .

In Case 1, just erase the actions a and w_i from $\Pi \xrightarrow{\nu} \Pi'$: let $\Pi \xrightarrow{\nu'} \Pi' = \Pi \xrightarrow{\nu_1} \Pi_1 \xrightarrow{\mu_{21}} \Pi_4 \xrightarrow{\nu_{22}} \Pi'$, where $\Pi_1 \xrightarrow{\mu_{21}} \Pi_4$ is (essentially) the same as $\Pi_2 \xrightarrow{\nu_{21}} \Pi_3$ except that the name of the conclusions on which the elementary actions apply have changed (since the action $\Pi_1 \xrightarrow{a} \Pi_2$ is no more present). Clearly, $\Pi \xrightarrow{\nu'} \Pi'$ is a $\neg\boxtimes$ -action and $?_c^0(\Pi \xrightarrow{\nu} \Pi') > ?_c^0(\Pi \xrightarrow{\nu'} \Pi')$.

In Case 2, erase the action a and slightly modify the action box_i : indeed, let $\Pi \xrightarrow{\nu'} \Pi'' = \Pi \xrightarrow{\nu_1} \Pi_1 \xrightarrow{\mu_{21}} \Pi'_3 \xrightarrow{\text{box}_{i-1}} \Pi'_4 \xrightarrow{\mu_{22}} \Pi''$, where

- $\Pi_1 \xrightarrow{\mu_{21}} \Pi'_3$ is the same as $\Pi_2 \xrightarrow{\nu_{21}} \Pi_3$ except that the name of the conclusions on which the elementary actions apply have changed (since the action $\Pi_1 \xrightarrow{a} \Pi_2$ is no more present);
- the set Π'_3 is Π_3 with one conclusion and one $?_c$ -cell with 0 inputs less in each of its elements;
- the action $\Pi'_3 \xrightarrow{\text{box}_{i-1}} \Pi'_4$ applies the rewrite rule in Figure 9i to the conclusion $i - 1$ of every element in Π'_3 , which corresponds to the conclusion i of every element in Π_3 ;
- the \boxtimes -cell of each element in Π'_4 created by $\Pi'_3 \xrightarrow{\text{box}_{i-1}} \Pi'_4$ has one output less than the \boxtimes -cell of each element in Π_4 created by $\Pi_3 \xrightarrow{\text{box}_i} \Pi_4$; the action $\Pi'_4 \xrightarrow{\mu_{22}} \Pi''$ and the set Π'' are obtained from $\Pi_4 \xrightarrow{\nu_{22}} \Pi'$ and Π' accordingly (in particular Π'' is \boxtimes -full).

Clearly, even in Case 2, $\Pi \xrightarrow{\nu'} \Pi''$ is a $\neg\boxtimes$ -action and $?_c^0(\Pi \xrightarrow{\nu} \Pi') > ?_c^0(\Pi \xrightarrow{\nu'} \Pi'')$. \square

By Lemmas 9.5 and 9.7, a set Π of DiLL_0 quasi-proof-structures is gluable if and only if it is canonically gluable without nullary $?_c$ -splittings. This restricts the search space for rewritings to decide if Π is gluable, making gluability a *decidable* problem in the finite case.

Proposition 9.8 (decidability of finite gluability). *The problem below is decidable.*

Data: a finite set Π of DiLL_0 quasi-proof-structures;

Question: is Π gluable?

Proof. We first define some notions that will be used in the proof. Akin to the size of a MELL quasi-proof-structure used in the proof of Proposition 6.8, the *size* $s(\Pi)$ of a finite set Π of DiLL_0 quasi-proof-structures is the quadruple (p, q, r, s) where:

- p is the finite multiset of the number of inputs of each $?$ -cell with at least one input in each element of Π (hence, p is a finite multiset of positive integers);
- q is the number of cells in Π different from $?$ -cells without inputs and from \bowtie -cells;
- r is the finite multiset $[k_1, \dots, k_n]$ where n is the number of conclusions in each element of Π and, for all $1 \leq i \leq n$, k_i is the number of $\rho \in \Pi$ whose conclusion i is the output of a $?$ -cell with at least one input;
- s is the finite multiset of the number of conclusions of each component of each element of Π .

Finite multisets are well-ordered as usual, quadruples are well-ordered lexicographically.

We say that a non- \bowtie -full set Π of DiLL_0 quasi-proof-structures *safely rewrites to* Π' if $\Pi \rightsquigarrow \Sigma \rightsquigarrow \Pi'$ for some set Σ , some scheduling ν only made of an arbitrary number (possibly 0) of exc_i such that $\Pi \rightsquigarrow \Sigma$ is a $\rightarrow\bowtie$ -action, and some elementary scheduling $a \neq \text{exc}_i$ such that $\Sigma \rightsquigarrow \Pi'$ is neither a \bowtie -action nor nullary $?$ -splitting.

For every finite set Π of DiLL_0 quasi-proof-structures, exactly one of the following holds:

- (1) either Π is \bowtie -full (this includes the case $\Pi = \{\bar{\varepsilon}\}$) and then it is gluable by Lemma 9.2;
- (2) or Π is not \bowtie -full, and every scheduling ν such that $\Pi \rightsquigarrow \Pi'$ is a $\rightarrow\bowtie$ -action for some Π' is only made of an arbitrary number — possibly 0 — of exc_i (this includes the case when no elementary scheduling applies to Π , in particular when two elements of Π have different types); then, by definition, Π is gluable if $\Pi = \emptyset$, and it is not gluable otherwise;
- (3) or Π is not \bowtie -full, and there are $h > 0$ sets Π_1, \dots, Π_h such that Π safely rewrites to Π_j for all $1 \leq j \leq h$, and if Π safely rewrites to Π' then $\Pi' = \Pi_j$ for some $1 \leq j \leq h$; then, Π is canonically gluable without nullary $?$ -splittings (which is equivalent to be gluable, by Lemmas 9.5 and 9.7) if and only if so is at least one among Π_1, \dots, Π_h .

Let us prove that when neither Case 1 nor Case 2 holds, we are in Case 3. If neither Case 1 nor Case 2 holds, Π is not \bowtie -full and $\Pi \rightsquigarrow \Sigma \rightsquigarrow \Pi'$ for some sets Σ, Π' , some scheduling ν only made of exc_i such that $\Pi \rightsquigarrow \Sigma$ is a $\rightarrow\bowtie$ -action, and some elementary scheduling $a \neq \text{exc}_i$ such that $\Sigma \rightsquigarrow \Pi'$ is not a \bowtie -action. If $a = ?_i$ and $\Sigma \rightsquigarrow \Pi'$ is nullary $?$ -splitting, then i is the output of a $?$ -cell in every element of Σ , and so there is a set Π'' such that $\Sigma \rightsquigarrow \Pi''$ is neither a \bowtie -action nor nullary $?$ -splitting, with $c \in \{?_i, ?_i, ?_i\}$ (if $c = ?_i$, $\Sigma \rightsquigarrow \Pi''$ just splits the inputs of the $?$ -cell with output i in some element of Σ in a way different from $\Sigma \rightsquigarrow \Pi'$). Therefore, Π safely rewrites to a finite set of DiLL_0 quasi-proof-structures. Clearly, it is impossible that Π safely rewrites to infinitely many sets, since each element of Π has a finite number of conclusions. Hence, Case 3 holds.

The *nondeterministic* procedure to decide if a finite set Π of DiLL_0 quasi-proof-structures is gluable consists of repeating the step described in Case 3, until all the sets Π_1, \dots, Π_n which Π rewrites to are of the form described in Case 1 or 2: Π is gluable if and only if so is at least one among Π_1, \dots, Π_n . The procedure terminates because (recall the size above):

- if $\Pi \xrightarrow{\text{exc}_i} \Pi'$ then $s(\Pi) = s(\Pi')$ but infinite chains of exc_i are irrelevant (see Footnote 27);
- if Π safely rewrites to Π' then $s(\Pi) > s(\Pi')$ (according to the lexicographic order). \square

The size used in the proof of Proposition 9.8 is well defined only if the set Π of DiLL_0 quasi-proof-structures is finite. The *finiteness* hypothesis in deciding gluability of Π is necessary, as we have seen in Remark 8.7: the infinite in depth Π shown there is not gluable and there are infinite rewritings from Π that are canonical and without nullary $?$ -splittings.

Theorem 9.9 (decidability). *The finite inverse Taylor expansion problem is decidable.*

Proof. Let Π be a finite set of DiLL_0 proof-structures without \boxtimes -cells. By Proposition 9.8, we can decide if Π is gluable. By Theorem 8.3, since there is no \boxtimes -cell in Π , this is equivalent to deciding if $\Pi \subseteq \mathcal{T}(R)$ for some MELL proof-structure R . \square

The nondeterministic procedure to decide the finite inverse Taylor expansion problem is given by the proof of decidability of finite gluability (Proposition 9.8) and is *constructive*: if a finite set of DiLL_0 proof-structures without \boxtimes -cells is included in the Taylor expansion $\mathcal{T}(R)$ of some MELL proof-structure R , our procedure allows us to construct one of these R .

10. NON-ATOMIC AXIOMS

Our (quasi-)proof-structures (Definitions 4.1 and 4.5, Figure 2) deal with *atomic axioms* only: the two outputs of any ax -cell are dual atomic formulas. We can relax the definitions and allow also the presence of *non-atomic axioms*: the outputs of any ax -cell are typed by dual MELL formulas, not necessarily atomic. We can extend our results to this more general setting, with some technical complications. The only trouble is with *exponential axioms*, where the outputs of an ax -cell are typed by MELL formulas of the form $!A^\perp, ?A$. Indeed, consider the daimons σ and σ' and the MELL proof-structure S' below.

$$\sigma = \begin{array}{c} \text{ax} \\ \swarrow \quad \searrow \\ !A^\perp \quad !A^\perp \quad ?A \end{array} \quad \sigma' = \begin{array}{c} \text{ax} \\ \swarrow \quad \searrow \\ !A^\perp \quad !A^\perp \quad ?A \quad ?A \end{array} \quad S' = \begin{array}{c} \text{ax} \\ \text{ax} \\ \swarrow \quad \searrow \\ !A^\perp \quad !A^\perp \quad ?A \quad ?A \end{array}$$

We have that $\{\sigma\} \xrightarrow{?_3} \{\sigma'\} \subseteq \mathcal{T}^{\boxtimes}(S')$. But no elementary scheduling $?_3$ can be applied backwards to S' (because of the lack of a $?_3$ -cell), which *breaks naturality* (Theorem 7.5). This is actually due to the fact that, with exponential axioms, co-functionality (Lemma 6.5) fails for the rewrite rule $?_i$: read from right to left, $?_i$ is a functional but not total relation.

The solution we propose here to deal with non-atomic axioms asks for a technical refinement of only few notions, the rest is unchanged and goes smoothly. The definitions of module, proof-structure and quasi-proof-structure (Definitions 4.1 and 4.5) change as follows.

- (1) In modules, non-atomic axioms are allowed, except the exponential ones. We add a new type of cells, the $!\text{ax}$ -cells, with no inputs and two outputs of type $!A^\perp$ and A .
- (2) In proof-structures (and hence in quasi-proof-structures), we require that, for every $!\text{ax}$ -cell whose outputs have type $!A^\perp$ and A , the output o of type A is an input of a $?_3$ -cell (more precisely, o is an half of an edge whose other half is the input of a $?_3$ -cell).

The requirement “atomic” in the elementary scheduling ax_i in Figure 7 is replaced by “non-exponential”, and we add the new elementary scheduling $!\text{ax}_i$ below to the list in Figure 7,

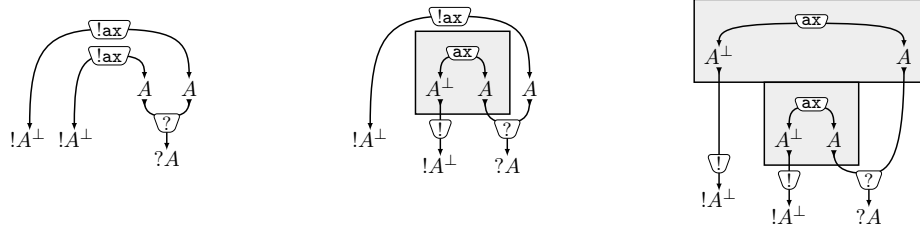
$(\Gamma_1; \dots; \Gamma_{k-1}; c(i), c(i+1); \Gamma_{k+1}; \dots; \Gamma_n) \xrightarrow{!\text{ax}_i} (\Gamma_1; \dots; \Gamma_{k-1}; \epsilon; \Gamma_{k+1}; \dots; \Gamma_n)$ if $c(i) = ?A = c(i+1)^\perp$ whose action—also in its daimoned version—is like the hypothesis in Figures 8c and 9c:



The way the action of an elementary scheduling is defined (Definitions 6.3 and 7.1) automatically rules out the possibility that a quasi-proof-structure rewrites to a module that is not a quasi-proof-structure. For instance, the elementary scheduling $?_d1$ does not apply to the (MELL and DiLL_0) proof-structure Q below, because it would yield a module Q' that is not a quasi-proof-structure (see Item 2 above).

$$Q = \begin{array}{c} \boxed{\text{!ax}} \\ \downarrow \\ A \quad !A^\perp \\ \downarrow \\ \text{?} \\ \downarrow \\ ?A \end{array} \xrightarrow{?_1} \begin{array}{c} \boxed{\text{!ax}} \\ \downarrow \\ A \quad !A^\perp \\ \downarrow \\ \text{?} \\ \downarrow \\ ?A \end{array} = Q'$$

Apart from the changes we mentioned, all definitions and claims in Sections 4 to 9 remain unaffected and valid (just add !ax_i to the list of elementary schedulings that can apply to R in Lemmas 6.6 and 6.7.2). The counterexample to naturality shown at the beginning of this section does not apply here, as S' is not a MELL quasi-proof-structure (because of its exponential axioms). In our framework with non-atomic axioms, the only MELL proof-structures containing σ' in their filled Taylor expansion are the ones below (up to the order of their conclusions), and the elementary scheduling $?_3$ applies backward to each of them.



The way we propose here to deal with non-atomic axioms is more elegant and less *ad hoc* than the one we presented in [GPT20, Section 7]. In the latter, we allowed the outputs of a \boxtimes -cell to be inputs of a $?$ -cell, and we changed the action of the elementary scheduling $?_i$ for DiLL_0 quasi-proof-structures in the daimoned case (Figure 9f), by explicitly requiring the presence of a $?$ -cells above the conclusion i . Consequently, in [GPT20] we had to redefine the filled Taylor expansion in the non-atomic case, and thus to recheck that naturality holds for *all* the rewrite rules. Here instead we do not have to redefine the filled Taylor expansion and we have to check naturality (Theorem 7.5) in the non-atomic case only for the action of !ax_i .

As another possible solution to deal with non-atomic axioms, instead of using generalized $?$ -cells in Definition 4.1 (with an arbitrary number of inputs of type A and one output of type $?A$, as in [DR95]), we could have used different kinds of $?$ -cells for dereliction (one input of type A , one output of type $?A$), contraction (two inputs of type $?A$, one output of type $?A$), and weakening (no inputs, one output of type $?A$), as in [Gir87]. Such a choice should be supported by a rework of the elegant definition of the Taylor expansion of a MELL proof-structure via the notion of pullback (Definitions 5.4 and 5.5), since it collapses dereliction, contraction and weakening in a generalized $?$ -cell.

11. CONCLUSIONS AND PERSPECTIVES

Daimons for empty boxes. Our gluability criterion (Theorem 8.3) solves the inverse Taylor expansion problem in an “asymmetric” way: we characterize the sets of DiLL_0 proof-structures without \boxtimes -cells that are included in the Taylor expansion of some MELL proof-structure, but in general DiLL_0 proof-structures might contain \boxtimes -cells (while MELL proof-structures cannot, see Definition 4.1). Daimons and emptyings are needed to get a natural transformation (via the *filled* Taylor expansion, Theorem 7.5), which is the main ingredient to prove our gluability criterion. But we are interested in frameworks without \boxtimes -cells. This asymmetry is

technically inevitable, due to the fact that a glueable set of DiLL_0 proof-structures might not contain any information on the content of some box, when they take 0 copies of it.

Comparison with Pagani and Tasson [PT09]. Pagani and Tasson’s solution [PT09] to the inverse Taylor expansion problem is less general than ours, because it characterizes *finite* sets of DiLL_0 proof-structures that are included in the Taylor expansion of some *cut-free MELL* proof-structure with *atomic axioms*. We do not have these limitations (finite, cut-free, atomic axioms), our characterization yields a decision procedure (Theorem 9.9) in the finite case as in [PT09], and we can smoothly adapt our criterion to the cut-free case (Theorem 8.3.2, which is not trivial as explained in Example 8.5). In [PT09], the restrictions to cut-free and to atomic axioms aim to simplify their presentation and might be overcome. But the restriction to *finite* sets of DiLL_0 proof-structures is somehow essential in their approach. Roughly, their machinery takes a finite set Π of DiLL_0 proof-structures as input, juxtaposes its elements in a unique graph π and then runs a rewriting by means of some tokens that go through the whole π and try to merge its components in a MELL proof-structure whose Taylor expansion includes Π , if any. If Π were an infinite set, π would be an infinite graph and, apart from the technical intricacies of dealing with infinite objects, it would be impossible to provide a characterization in that case. In particular, their approach could not distinguish whether Π is infinite in width (which can be in the Taylor expansion of some MELL proof-structure) or infinite in depth (which cannot, see Remark 8.7). Our approach, instead, defines a rewrite relation on a single—finite—element of Π and extends it to the whole (possibly infinite) Π by requiring that the same rewrite rule applies to each element of Π (Definition 7.1). Thus, we can accommodate the case where Π is infinite.

We believe that our rewriting rules are also simpler than the ones in [PT09] and rely on a more abstract and less *ad hoc* property (naturalness), that allows us to prove also semi-decidability of another problem: *type inhabitation* for cut-free MELL proof-structures.

Finally, Pagani and Tasson’s solution of the inverse Taylor expansion problem is affected by another limitation, even though not particularly emphasized in [PT09]: their Theorem 2 (analogous to the “only if” part of our Theorem 8.3) assumes not only that their rewriting starting from a set of DiLL_0 proof-structures terminates but also that it ends on a MELL proof-structure, according to their definition of MELL proof-structure. This is limiting when in the set of DiLL_0 proof-structures there is no information about the content of a box. For instance, consider the singletons Π and Π' of DiLL_0 proof-structures below.

$$\Pi = \left\{ \begin{array}{c} \Downarrow \\ !1 \end{array} \right\} \quad \Pi' = \left\{ \begin{array}{c} \Downarrow \\ !X \end{array} \right\} \quad R = \begin{array}{c} \boxed{\Downarrow} \\ \downarrow \\ \Downarrow \\ !1 \end{array}$$

Pagani and Tasson’s rewrite rules do not distinguish the two singletons, each one is included in the Taylor expansion of cut-free MELL proof-structures with an “empty box”, due to the lack of information. So their notion of MELL proof-structure is wider and non-standard (because it allows the presence of empty boxes). On the contrary, our rewrite rules distinguish Π and Π' : via the action of cut-free schedulings, the former can rewrite to $\{\varepsilon\}$ and is included in the Taylor expansion of the MELL proof-structure R above, the latter cannot rewrite to $\{\varepsilon\}$ and is not part of the Taylor expansion of any cut-free MELL proof-structure. Summing up, our characterization follows the standard notion of MELL proof-structures (unlike [PT09]) and is more fine-grained and informative than the one in [PT09].

The λ -calculus, connectedness and coherence. Our rewriting system and gluability criterion might help to prove that a binary coherence relation can solve the inverse Taylor expansion problem for MELL proof-structures fulfilling some geometric property related to connectedness, despite the impossibility for the full MELL fragment. Such a coherence would extend the coherence criterion for resource λ -terms. Note that our gluability criterion is actually an extension of the criterion for resource λ -terms. Indeed, in the case of the λ -calculus, there are three rewrite steps, corresponding to abstraction, application and variable (which can be encoded in our rewrite steps), and coherence is defined inductively: if a set of resource λ -terms is coherent, then any set of resource λ -term that rewrites to it is also coherent.

Presented in this way, the main difference between the λ -calculus and “connected” MELL (concerning the inverse Taylor expansion problem) would not be because of the rewriting system, but because the structure of any resource λ -term uniquely determines the rewriting path, while for DiLL₀ proof-structures we have to quantify existentially over all possible paths. This is an unavoidable consequence of the fact that proof-structures do not have a tree-structure, contrary to λ -terms.

Moreover, it is possible to match and mix different rewritings. Indeed, consider three DiLL₀ proof-structures pairwise gluable: proving that they are gluable as a whole amounts to computing a rewriting from the three rewritings witnessing their pairwise gluability. Our rewriting system has been designed with that mixing-and-matching operation in mind, in the particular case where the boxes are connected. This is reminiscent of [GPT16], where we also showed that a certain property enjoyed by the λ -calculus can be extended to proof-structures, provided they are connected inside boxes. We leave it as future work.

Functoriality and naturality. Our functorial point of view on proof-structures might unify many results. Let us cite two of them.

- A sequent calculus proof of $\vdash \Gamma$ can be translated to a path from the empty sequent to Γ . This could be the starting point for the formulation of a new correctness criterion.
- The category **Sched** can be extended with a higher structure—transforming it from a category into a 2-category—which allows cut-elimination to be represented as a 2-arrow. The functors \mathfrak{qMELL} and \mathfrak{PqDiLL}_0 can also be extended to 2-functors, so as to prove via naturality that cut-elimination and the Taylor expansion commute.

Acknowledgments. The authors are grateful to Olivier Laurent, Lionel Vaux and the anonymous reviewers for their insightful comments.

REFERENCES

- [Béc98] Denis Bécet. Minimality of the correctness criterion for multiplicative proof nets. *Mathematical Structures in Computer Science*, 8(6):543–558, 1998.
- [BHP13] Pierre Boudes, Fanny He, and Michele Pagani. A characterization of the Taylor expansion of lambda-terms. In *Computer Science Logic 2013 (CSL 2013)*, volume 23 of *LIPICs*, pages 101–115. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013. doi:10.4230/LIPICs.CSL.2013.101.
- [BM07] Dennis V. Borisov and Yuri I. Manin. *Generalized Operads and Their Inner Cohomomorphisms*, volume 265 of *Progress in Mathematics*, pages 247–308. Birkhäuser Basel, Basel, 2007. doi:10.1007/978-3-7643-8608-5_4.
- [BM20] Davide Barbarossa and Giulio Manzonetto. Taylor subsumes Scott, Berry, Kahn and Plotkin. *Proc. ACM Program. Lang.*, 4(POPL):1:1–1:23, 2020. doi:10.1145/3371069.

- [Bou93] Gérard Boudol. The lambda-calculus with multiplicities (abstract). In *4th International Conference on Concurrency Theory (CONCUR '93)*, volume 715 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 1993. doi:10.1007/3-540-57208-2_1.
- [Bou09] Pierre Boudes. Thick subtrees, games and experiments. In *Typed Lambda Calculi and Applications, 9th International Conference (TLCA 2009)*, volume 5608 of *Lecture Notes in Computer Science*, pages 65–79. Springer, 2009. doi:10.1007/978-3-642-02273-9_7.
- [CV18] Jules Chouquet and Lionel Vaux. An application of parallel cut elimination in unit-free multiplicative linear logic to the taylor expansion of proof nets. In *27th EACSL Annual Conference on Computer Science Logic, CSL 2018*, volume 119 of *LIPICs*, pages 15:1–15:17. Schloss Dagstuhl, 2018. doi:10.4230/LIPICs.CSL.2018.15.
- [dC16] Daniel de Carvalho. The relational model is injective for multiplicative exponential linear logic. In *25th Annual Conference on Computer Science Logic (CSL 2016)*, volume 62 of *LIPICs*, pages 41:1–41:19. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPICs.CSL.2016.41.
- [dC18] Daniel de Carvalho. Taylor expansion in linear logic is invertible. *Logical Methods in Computer Science*, 14(4), 2018. doi:10.23638/LMCS-14(4:21)2018.
- [dCT12] Daniel de Carvalho and Lorenzo Tortora de Falco. The relational model is injective for multiplicative exponential linear logic (without weakenings). *Annals of Pure and Applied Logic*, 163(9):1210–1236, 2012. doi:10.1016/j.apal.2012.01.004.
- [DPS21] Abhishek De, Luc Pellissier, and Alexis Saurin. Canonical proof-objects for coinductive programming: infinets with infinitely many cuts. In *23rd International Symposium on Principles and Practice of Declarative Programming (PPDP 2021)*, pages 7:1–7:15. ACM, 2021.
- [DR95] Vincent Danos and Laurent Regnier. Proof-nets and the Hilbert Space. In *Proceedings of the Workshop on Advances in Linear Logic*, pages 307–328. Cambridge University Press, 1995.
- [DS19] Abhishek De and Alexis Saurin. Infinets: The parallel syntax for non-wellfounded proof-theory. In *TABLEAUX 2019 - 28th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, volume 11714 of *Lecture Notes in Computer Science*. Springer, 2019. doi:10.1007/978-3-030-29026-9_17.
- [Ehr02] Thomas Ehrhard. On Köthe sequence spaces and linear logic. *Mathematical Structures in Computer Science*, 12(5):579–623, 2002. doi:10.1017/S0960129502003729.
- [Ehr05] Thomas Ehrhard. Finiteness spaces. *Mathematical Structures in Computer Science*, 15(4):615–646, 2005. doi:10.1017/S0960129504004645.
- [ER03] Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1-3):1–41, 2003. doi:10.1016/S0304-3975(03)00392-X.
- [ER06a] Thomas Ehrhard and Laurent Regnier. Böhm Trees, Krivine’s Machine and the Taylor Expansion of Lambda-Terms. In *Second Conference on Computability in Europe (CiE 2006)*, volume 3988 of *Lecture Notes in Computer Science*, pages 186–197. Springer, 2006. doi:10.1007/11780342_20.
- [ER06b] Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theoretical Computer Science*, 364(2):166–195, 2006. doi:10.1016/j.tcs.2006.08.003.
- [ER08] Thomas Ehrhard and Laurent Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. *Theoretical Computer Science*, 403(2-3):347–372, 2008. doi:10.1016/j.tcs.2008.06.001.
- [Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987. doi:10.1016/0304-3975(87)90045-4.
- [GPT16] Giulio Guerrieri, Luc Pellissier, and Lorenzo Tortora de Falco. Computing connected proof(-structure)s from their Taylor expansion. In *1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016)*, volume 52 of *LIPICs*, pages 20:1–20:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPICs.FSCD.2016.20.
- [GPT19] Giulio Guerrieri, Luc Pellissier, and Lorenzo Tortora de Falco. Proof-net as graph, Taylor expansion as pullback. In *Logic, Language, Information, and Computation - 26th International Workshop (WoLLIC 2019)*, volume 11541 of *Lecture Notes in Computer Science*, pages 282–300. Springer, 2019. doi:10.1007/978-3-662-59533-6_18.
- [GPT20] Giulio Guerrieri, Luc Pellissier, and Lorenzo Tortora de Falco. Glueability of resource proof-structures: inverting the Taylor expansion. In *28th EACSL Annual Conference on Computer Science Logic, CSL 2020*, volume 152 of *LIPICs*, pages 24:1–24:18. Schloss Dagstuhl, 2020. doi:10.4230/LIPICs.CSL.2020.24.

- [Kan94] Max I. Kanovich. The complexity of Horn fragments of linear logic. *Ann. Pure Appl. Log.*, 69(2-3):195–241, 1994. doi:10.1016/0168-0072(94)90085-X.
- [KMP20] Emma Kerinec, Giulio Manzonetto, and Michele Pagani. Revisiting call-by-value Böhm trees in light of their Taylor expansion. *Log. Methods Comput. Sci.*, 16(3), 2020. doi:10.23638/LMCS-16(3:6)2020.
- [Laf90] Yves Lafont. Interaction nets. In *Seventeenth Annual ACM Symposium on Principles of Programming Languages (POPL 1990)*, pages 95–108. ACM Press, 1990. doi:10.1145/96709.96718.
- [LMSS92] Patrick Lincoln, John Mitchell, Andre Scedrov, and Natarajan Shankar. Decision problems for propositional linear logic. *Annals of Pure and Applied Logic*, 56(1):239 – 311, 1992. doi:10.1016/0168-0072(92)90075-B.
- [MP07] Damiano Mazza and Michele Pagani. The separation theorem for differential interaction nets. In *Logic for Programming, Artificial Intelligence, and Reasoning, 14th International Conference (LPAR 2007)*, volume 4790 of *Lecture Notes in Computer Science*, pages 393–407. Springer, 2007. doi:10.1007/978-3-540-75560-9_29.
- [MPV18] Damiano Mazza, Luc Pellissier, and Pierre Vial. Polyadic approximations, fibrations and intersection types. *PACMPL*, 2(POPL):6:1–6:28, 2018. doi:10.1145/3158094.
- [MR14] Giulio Manzonetto and Domenico Ruoppolo. Relational graph models, Taylor expansion and extensionality. *Electronic Notes in Theoretical Computer Science*, 308:245–272, 2014. doi:10.1016/j.entcs.2014.10.014.
- [Pag09] Michele Pagani. The cut-elimination theorem for differential nets with promotion. In *Typed Lambda Calculi and Applications, 9th International Conference, (TLCA 2009)*, volume 5608 of *Lecture Notes in Computer Science*, pages 219–233. Springer, 2009. doi:10.1007/978-3-642-02273-9_17.
- [Par92] Michel Parigot. $\lambda\mu$ -calculus: An algorithmic interpretation of classical natural deduction. In *Logic Programming and Automated Reasoning, International Conference (LPAR '92)*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer, 1992. doi:10.1007/BFb0013061.
- [PT09] Michele Pagani and Christine Tasson. The inverse Taylor expansion problem in linear logic. In *24th Annual Symposium on Logic in Computer Science (LICS 2009)*, pages 222–231. IEEE Computer Society, 2009. doi:10.1109/LICS.2009.35.
- [PTV16] Michele Pagani, Christine Tasson, and Lionel Vaux. Strong normalizability as a finiteness structure via the Taylor expansion of λ -terms. In *Foundations of Software Science and Computation Structures - 19th International Conference (FOSSACS 2016)*, volume 9634 of *Lecture Notes in Computer Science*, pages 408–423. Springer, 2016. doi:10.1007/978-3-662-49630-5_24.
- [Ret97] Christian Retoré. A semantic characterisation of the correctness of a proof net. *Mathematical Structures in Computer Science*, 7(5):445–452, 1997. doi:10.1017/S096012959700234X.
- [Tas09] Christine Tasson. *Sémantiques et Syntaxes Vectorielles de la Logique Linéaire*. PhD thesis, Université Paris Diderot, France, December 2009. URL: <https://tel.archives-ouvertes.fr/tel-00440752>.
- [TdF03] Lorenzo Tortora de Falco. Additives of linear logic and normalization - part I: a (restricted) church-rosser property. *Theor. Comput. Sci.*, 294(3):489–524, 2003. doi:10.1016/S0304-3975(01)00176-1.
- [Tra09] Paolo Tranquilli. Confluence of pure differential nets with promotion. In *Computer Science Logic, 23rd international Workshop, CSL 2009, 18th Annual Conference of the EACSL*, volume 5771 of *Lecture Notes in Computer Science*, pages 500–514. Springer, 2009. doi:10.1007/978-3-642-04027-6_36.
- [Tra11] Paolo Tranquilli. Intuitionistic differential nets and lambda-calculus. *Theor. Comput. Sci.*, 412(20):1979–1997, 2011. doi:10.1016/j.tcs.2010.12.022.
- [Vau17] Lionel Vaux. Taylor expansion, lambda-reduction and normalization. In *26th EACSL Annual Conference on Computer Science Logic (CSL 2017)*, volume 82 of *LIPICs*, pages 39:1–39:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPICs.CSL.2017.39.