Relational type-checking for MELL proof-structures. Part 1: Multiplicatives

Giulio Guerrieri
Institut de Mathématiques de Marseille, UMR 7373, Aix-Marseille Université, Centrale Marseille
F-13453 Marseille, France
giulio.guerrieri@univ-amu.fr

Luc Pellissier
LIPN, UMR 7030, Université Paris 13, Sorbonne Paris Cité
F-93430 Villetaneuse, France
luc.pellissier@lipn.univ-paris13.fr

Lorenzo Tortora de Falco
Dipartimento di Matematica e Fisica, Università Roma Tre, Rome, Italy
tortora@uniroma3.it

Relational semantics for linear logic is a form of non-idempotent intersection type system, from which several informations on the execution of a proof-structure can be recovered. An element of the relational interpretation of a proof-structure R with conclusion Γ acts thus as a type (refining Γ) having R as an inhabitant.

We are interested in the following type-checking question: given a proof-structure R, a list of formulæ Γ, and a point x in the relational interpretation of Γ, is x in the interpretation of R? This question is decidable. We present here an algorithm that decides it in time linear in the size of R, if R is a proof-structure in the multiplicative fragment of linear logic. This algorithm can be extended to larger fragments of multiplicative-exponential linear logic containing λ-calculus.

1 Introduction

Intersection types have been introduced as a way of extending the λ-calculus' simple types to finite polymorphism, by adding a new type constructor $\cap$ and new typing rules governing it. Contrarily to simple types (which are sound but incomplete), intersection types present a sound and complete characterization of strong normalization.

Intersection types were first formulated as idempotent, that is, they verify the equation $\alpha \cap \alpha = \alpha$. This corresponds to an interpretation of a typed term $M : \alpha \cap \beta$ as *M can be used both as data of type α and as data of type β*. In a non-idempotent setting (*i.e.* by dropping the equation $\alpha \cap \alpha = \alpha$), the meaning of the typed term $M : \alpha \cap \alpha \cap \beta$ is refined as *M can be used twice as data of type α and once as data of type β*. Non-idempotent intersection types have been used to get qualitative and quantitative information on the execution time of λ-terms [2, 9].

One of the simplest denotational model of Linear Logic (LL, [7]) is relational semantics, *i.e.* the $*$-autonomous category **Rel** of sets and relations endowed with the co-monad ! of finite multisets. In this setting, a LL formula is interpreted by a set, and a LL proof-structure[1] by a relation between sets. The

[1]Following [7], we make a difference between proof-structures and proof-nets: a proof-net is a proof-structure corresponding to a derivation in LL sequent calculus. Proof-nets can be characterized among proof-structures via geometric correctness criteria.

Submitted to:
ITRS 2016

relational semantics $\mathbf{Rel}_!$ of the λ-calculus is just the co-Kleisli category constructed from the co-monad !. The syntactic counterpart of this construction is Girard's encoding of intuitionistic logic into LL.

Relational semantics corresponds to a non-idempotent intersection type system, called System R in [2] (see also [17]), for LL and λ-calculus. We sketch the interpretation in System R of simply typed λ-terms having o as unique base simple type. Non-idempotent intersection types of System R has to be seen as a refinement of simple types, the reader has to pay attention to distinguish the two sorts of types.

Let $\mathscr{A}t$ be a set of so-called atoms (*i.e.* the atomic types of System R). With every simple type σ, we associate a set $|\sigma|$ (intuitively, the set of non-idempotent intersection types associated with σ) as follows:

$$|o| = \mathscr{A}t \qquad\qquad |\sigma \to \tau| = \mathscr{M}_{\text{fin}}(|\sigma|) \times |\tau|,$$

where $\mathscr{M}_{\text{fin}}(\cdot)$ denotes the set of finite multisets. We denote finite multisets by square bracket, so that $[]$ denotes the empty multiset; $[\alpha, \alpha, \beta]$ denotes a multiset having two occurrences of the element α and one occurrence of β. Finite multisets are the natural setting to interpret a connective $\cap$ that is associative and commutative but not idempotent: if α and β are non-idempotent intersection types, then $[\alpha, \alpha, \beta]$ represents the non-idempotent intersection type $\alpha \cap \alpha \cap \beta$. We will at times write $X \to \alpha$ as a semantically-flavoured notation for the pair $(X, \alpha) \in |\sigma \to \tau|$.

With every valid simply-typing sequent $\vec{x} : \vec{\sigma} \vdash M : \tau$ (where $\vec{x} : \vec{\sigma} = x_1 : \sigma_1, \ldots, x_n : \sigma_n$), we associate its interpretation in the relational semantics (to be understood as the set of non-idempotent intersection type judgments associated with $\vec{x} : \vec{\sigma} \vdash M : \tau$), *i.e.* a set

$$[\![\vec{x} : \vec{\sigma} \vdash M : \tau]\!] \subseteq \mathscr{M}_{\text{fin}}(|\sigma_1|) \times \cdots \times \mathscr{M}_{\text{fin}}(|\sigma_n|) \times |\tau|$$

defined by induction on the simple type derivation $\vec{x} : \vec{\sigma} \vdash M : \tau$ (see [2] for details). In this way, the relational semantics of a closed λ-term is just the set of its non-idempotent intersection types. For a closed term M, we will write $\rhd M : \alpha : \sigma$ for $\alpha \in [\![\vdash M : \sigma]\!]$, emphasizing that the non-idempotent intersection type α refines the simple type σ. In general, we speak indifferently of non-idempotent intersection types or points in the relational interpretation of a λ-term.

As $\mathbf{Rel}_!$ is a cartesian closed category, $M =_\beta N$ implies $[\![\vec{x} : \vec{\sigma} \vdash M : \tau]\!] = [\![\vec{x} : \vec{\sigma} \vdash N : \tau]\!]$. Relational semantics provides also a denotational model for the untyped λ-calculus, since it is easy to build a reflexive object in $\mathbf{Rel}_!$, see [2].

We now give examples of the kind of information that can be recovered from the relational semantics:

- Let $\mathbf{B} = o \to o \to o$, the simple type for Church booleans. Let **true** $= \lambda xy.x$ and **false** $= \lambda xy.y$. We have $\rhd\,$**true** $: [*] \to [] \to * : \mathbf{B}$ but not $\rhd\,$**false** $: [*] \to [] \to * : \mathbf{B}$, where $* \in |o|$. As a consequence:

  Theorem 1. *Let M be a closed term of type* $\mathbf{B}$. *Then, $M =_\beta$* **true** *iff* $\rhd M : [*] \to [] \to * : \mathbf{B}$.

- More generally, a λ-term M has a sort of principal non-idempotent intersection type: its 1-point (obtained by taking only multisets of cardinality 1 in positive position: it can be computed efficiently when M is in normal form). That is, given a λ-term M of type σ in normal form, let M be its 1-point; then, for any λ-term N of simple type σ, $M =_\beta N$ if and only if $\rhd N : \mathsf{M} : \sigma$.

- Intersection types based on a variant of relational semantics have been shown useful [10] to encode verification problems.

- A variant of relational semantics, Scott model, can be used as a faster alternative to β-evaluation [18].

- Given two untyped normal forms N_1 and N_2, from the relational semantics of N_1N_2 it is possible to determine if N_1N_2 is normalizable and, in that case, to compute the length of the (most long) reduction of N_1N_2 to its normal form [2, 9] (see also [5, 4] for analogous results in the setting of linear logic proof-structures).

Such information becomes valuable when it is easy to determine whether a point belongs to the relational interpretation of a λ-term. In other words, we are interested in the tractability of the following decision problem about type-checking: given a relational point x and a λ-term M, can M be typed by x?

Instead of addressing this question directly in the λ-calculus, we tackle this study on the most general setting of LL proof-structures. Indeed, the aforementioned Girard's encoding, based on the call-by-name translation of the intuitionistic arrow $\sigma \to \tau$ into $!\sigma \multimap \tau$, embeds the simply typed λ-calculus into the multiplicative-exponential fragment (MELL) of LL, and this embedding extends to the untyped λ-calculus by adding the recursive types identity $o = o \to o$, *i.e.* $o = !o \multimap o$.

As a first step towards the resolution of this type-checking problem, we restrict ourselves to proof-structures in *multiplicative* LL (MLL), a fragment of MELL. In MLL, differently from MELL, the relational interpretation of a formula is finite (up to innocuous renaming). We aim to climb in the ladder of several larger fragments of MELL, providing algorithms of increasing complexity deciding this problem.

This kind of type-checking problems have been present since the dawn of LL. In its seminal article, Girard [7, §3.16, Remark (ii), p. 57] addresses the decidability issue of the following question: given a proof π (of conclusion Γ) and a point x (in the coherent interpretation of Γ), is x in the coherent interpretation[2] of π? The coherent setting is very different from the relational setting; indeed, unlike relational semantics, coherent semantics is not able to distinguish between certain (non-connected) MELL proof-structures [19, 6, 11, 3]. Nonetheless, we note along with Girard that the type-checking problem – in the relational setting – is trivial for MLL proof-structures *without cuts*: indeed, it suffices to propagate the information present in the conclusions of a MLL proof-structure. On the contrary, cuts allow to hide certain parts of the proof-structure from its conclusions (see Figure 4); in the presence of cuts, cycles need a special treatment. In this article, we will introduce a general framework deciding the relational type-checking problem for MLL proof-structures, possibly *with cuts*.

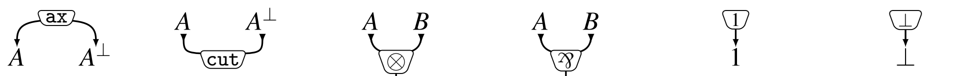
This framework offers an interactive formulation of intersection typing (and thus semantical evaluation). By this, we mean that it enjoys a certain closeness with the Geometry of Interaction machines. The Geometry of Interaction can be thought as a compilation of proof-structures and terms into automata [1]. These automata can then be optimized in a way that yields machines isomorphic to well-known abstract machines (such as the Krivine Abstract Machine or the Pointer Abstract Machine) in the cases of translations of the λ-calculus. As such, the Geometry of Interaction provides an unified framework for reasoning with λ-calculus abstract machines. We advocate here that the *computation flow* intuition captured in the Geometry of Interaction can be used to reason efficiently about semantic evaluation.

We think that this unification can be very fruitful: to be efficient, semantic evaluation is mixed with β-reduction, handled through mechanisms that look a lot like abstract machines [18]. We believe that the unification we tackle here allows for a fine-grained understanding of the evaluation of λ-calculus and MELL proof-structures.

We will define a variant of Vector Addition Systems (VAS, see [15]) that encode naturally our decision problem. Our algorithm bears a close resemblance with the Interaction Abstract Machine (see for instance [14]). It has indeed been known for a long time in the LL community that Geometry of Interaction and relational semantics enjoy a certain closeness. This work aims to bridge them on the operational side. It is moreover possible that this could give an enlightening illustration of the Int construction of [13] transforming a traced monoidal category (and thus a model of the Geometry of Interaction) into a compact-close category (and thus a denotational model of MELL).

We provide an algorithm that decides in time linear in the size of the MLL proof-structure R (with conclusion Γ) whether a point x (in the relational interpretation of Γ) is in the relational interpretation of R

[2]The coherent semantics is the first denotational model of LL. Relational semantics can be seen as a simplification of it.



(a) Axiom (b) Cut (c) Tensor (d) Par (e) One (f) Bottom

Figure 1: The cells for MLL.

(*relational type-checking problem for* MLL *proof-structures*). We give indications on how this algorithm can be extended in a bilinear (in the size of the term and of the point) algorithm in the case of λ-terms. For more and more general proof-structures, the complexity keeps increasing: indeed, box-connected MELL proof-structures (a geometric condition introduced and studied in [11]) require local non-determinism, while full MELL proof-structures require global non-determinism.

2 Elements of MLL syntax

The set of MLL *formulas* is generated by the grammar:

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

where X ranges over an infinite countable set of *propositional variables*. The linear negation $A^{\perp}$ of a formula A is involutive, *i.e.* $A^{\perp\perp} = A$, and defined via De Morgan laws $1^{\perp} = \perp$ and $(A \otimes B)^{\perp} = A^{\perp} \parr B^{\perp}$. If $\Gamma = (A_1, \dots, A_n)$ is a finite sequence of MLL formulas (with $n \in \mathbf{N}$), then $\parr\Gamma = A_1 \parr \cdots \parr A_n$; in particular, if $n = 0$ then $\parr\Gamma = \perp$.

Proof-structures offer a syntax for a graphical representation of MLL proofs. MLL proof-structures are directed labelled graphs Φ built from the *cells* (*i.e.* nodes labelled by a symbol in $\{1, \perp, \otimes, \parr, ax, cut\}$ called *type*) defined in Figure 1. We call *ports* the directed edges of such graphs, labelled by MLL formulas. For every cell, its ports are divided into *principal ports* (outgoing in the cell, depicted down in the picture) and *auxiliary ports* (incoming in the cell, depicted up). Any port p of a MLL proof-structure Φ is principal for exactly one cell and auxiliary for at most one cell; the MLL formula labelling p is denoted by $\mathsf{tp}_{\Phi}(p)$.

Let Φ be a MLL proof-structure. We denote by $\mathscr{P}(\Phi)$ the set of its ports and $\mathscr{C}(\Phi)$ the set of its cells. Let c be a cell of Φ. We denote by:

- $\mathsf{P}^{\mathsf{pri}}_{\Phi}(c)$ the principal ports of c. It is either
  - a single port, if c is of type $1, \perp, \otimes, \parr$;
  - an ordered pair of ports $\langle p_1, p_2 \rangle$, if c is of type ax;
  - empty, if c is of type cut;
- $\mathsf{P}^{\mathsf{aux}}_{\Phi}(c)$ the auxiliary ports of c. It is either
  - empty, if c is of type $ax, 1, \perp$;
  - an ordered pair of ports $\langle p_1, p_2 \rangle$, if c is of type cut, $\parr$ or $\perp$.

In a MLL proof-structure Φ, any port that is principal for some cell but is not auxiliary for any cell of Φ is called a *free port* of Φ. We will only consider in the sequel MLL proof-structures with a fixed (total) order on their free ports. Given a list of MLL formulæ $\Gamma = (A_1, \dots, A_n)$ with $n \in \mathbf{N}$, we say that an MLL proof-structure Φ is of *conclusion* Γ if the free ports of Φ are the ordered sequence $p_1 < \cdots < p_n$ and $\mathsf{tp}_{\Phi}(p_i) = A_i$ for every $i \in \{1, \dots, n\}$.

The MLL proof-structure R in Figure 2a has two free ports, one on the far right, the other on the far left. We will always depict MLL proof-structures with free ports ordered from left to right, so R in Figure 2a is of conclusion $(A \otimes B, A^\perp \parr B^\perp)$.

The definition of the cut-elimination process for MLL proof-structures is out of the scope of this paper, see [7] for details. We just point out that in the MLL fragment the cut-elimination is finitary and local: it is strongly normalizing and cannot duplicate or discard resources, which can thus be used only linearly.

3 Elements of relational semantics

We introduce here a variant of relational semantics (the simplest semantics of LL, where formulæ are interpreted by sets, and proof-structures by relations between sets) parametrized by a set $\mathscr{V}$ of variables. In this variant, thanks to variables in $\mathscr{V}$, parts of a relational point can be left “uninterpreted”, allowing for unification. We will use this feature in Section 4 to deal with cuts and cycles.

Definition 1 (Web of a MLL formula)**.** Let $\mathscr{A}t$ be a countably infinite set that does not contain any symbols in $\{1, \perp, \otimes, \parr, ax, cut\}$, any pairs or the empty sequence $()$; the elements of $\mathscr{A}t$ are called *atoms*.

Let $\mathscr{V}$ be a countably infinite set, disjoint from $\mathscr{A}t$, whose elements are the (*atomic*) *variables*.

The relational interpretation of MLL formulas is given by a function $|\cdot|_\mathscr{V}$ defined inductively by:

$$\begin{aligned} |X^\perp|_\mathscr{V} &= |X|_\mathscr{V} = \mathscr{A}t \cup \mathscr{V}, \text{ for all propositional variable } X; \\ |1|_\mathscr{V} &= |\perp|_\mathscr{V} = \{()\}; \\ |A \otimes B|_\mathscr{V} &= |A \parr B|_\mathscr{V} = (|A|_\mathscr{V} \times |B|_\mathscr{V}) \cup \mathscr{V}, \end{aligned}$$

For any MLL formula A, the set $|A|_\mathscr{V}$ is called the *web of* A, whose elements are the (*relational*) *points of* A. The set of relational points is $\mathbb{REL} = \bigcup_A |A|_\mathscr{V}$, where A ranges over all MLL formulæ.

The usual relational web of a MLL formula is recovered as $|\cdot|_\varnothing$, *i.e.* taking $\mathscr{V} = \varnothing$. We fix from now on a countably infinite set $\mathscr{V}$ of variables. Note that, for any MLL formula A, one has $|A|_\mathscr{V} = |A^\perp|_\mathscr{V}$.

Getting back to the parallel between relational semantics and non-idempotent intersection type systems, the relational interpretation $|A|_\mathscr{V}$ of a MLL formula A can be seen as the set of non-idempotent intersection types associated with A.

We define relational experiments straightforwardly on MLL proof-structures by adapting the definition in [7]. Given a MLL proof-structure Φ, a partial experiment of Φ is a labelling of some ports of Φ with relational points, “coherently” with the structure of Φ.

Definition 2 (Experiment of a MLL proof-structure)**.** Let Φ be a MLL proof-structure.

A *partial experiment* e of Φ is a partial function associating with $p \in \mathscr{P}(\Phi)$ an element of $|\mathsf{tp}_\Phi(p)|_\mathscr{V}$ verifying the following conditions, if e is defined on all the mentioned ports: let c be a cell in $\mathscr{C}(\Phi)$,

- if c is of type ax with $\mathsf{P}^{\mathsf{pri}}_\Phi(c) = \langle p, q \rangle$, then $\mathsf{e}(p) = \mathsf{e}(q)$;
- if c is of type cut with $\mathsf{P}^{\mathsf{aux}}_\Phi(c) = \langle p, q \rangle$, then $\mathsf{e}(p) = \mathsf{e}(q)$;
- if c is of type 1 or $\perp$ with $\mathsf{P}^{\mathsf{pri}}_\Phi(c) = q$, then $\mathsf{e}(q) = ()$;
- if c is of type $\otimes$ or $\parr$ with $\mathsf{P}^{\mathsf{aux}}_\Phi(l) = \langle p_1, p_2 \rangle$ and $\mathsf{P}^{\mathsf{pri}}_\Phi(l) = q$, then $\mathsf{e}(q) = (\mathsf{e}(p_1), \mathsf{e}(p_2))$.

An *experiment* is a partial experiment defined on all ports of Φ, whose codomain can be restricted to $\bigcup_{p \in \mathscr{P}(\Phi)} |\mathsf{tp}_\Phi(p)|_\varnothing$.

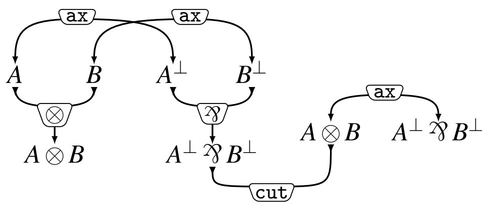
If we consider the Cartesian product of sets to be literally associative, given a MLL proof-structure Φ of conclusion $\Gamma = (A_1, \dots, A_n)$ (with ordered free ports $p_1 < \dots < p_n$), an experiment e of Φ defines naturally its *result* $|\mathsf{e}|$ as the relational point $(\mathsf{e}(p_1), \dots, \mathsf{e}(p_n)) \in |\parr\Gamma|_\varnothing$.

The *relational interpretation* of a MLL proof-structure Φ is then $[\![\Phi]\!] = \{|\mathsf{e}| : \mathsf{e} \text{ experiment of } \Phi\}$.

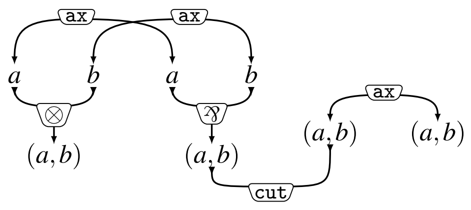
The reader can easily check that the relational semantics for MLL proof-structures is invariant under cut elimination: if $\Phi \rightsquigarrow_{\mathsf{cut}} \Phi'$ then $[\![\Phi]\!] = [\![\Phi']\!]$.

If we see the relational semantics as a non-idempotent intersection type system, an experiment of a MLL proof-structure is a type derivation, and its result is the conclusion of this type derivation. Just as we did for the λ-calculus in §1, given a MLL proof-structure R of conclusion Γ, we write $\rhd R : \alpha : \Gamma$ for $\alpha \in [\![R]\!] \subseteq |\parr\Gamma|_\varnothing$. The point α acts both as a witness of the fact that Γ types R, while refining this type.

The function depicted in Figure 2b is an experiment e of the MLL proof-structure of Figure 2a, where a and b are atoms. The experiment e proves $\rhd R : ((a,b),(a,b)) : (A \otimes B) \parr (A^\perp \parr B^\perp)$.



(a) The MLL proof-structure R

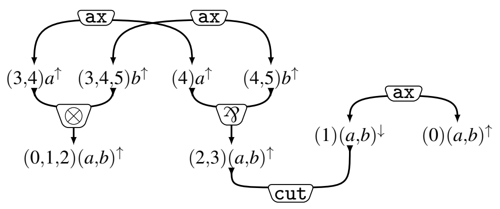


(b) The experiment e of R

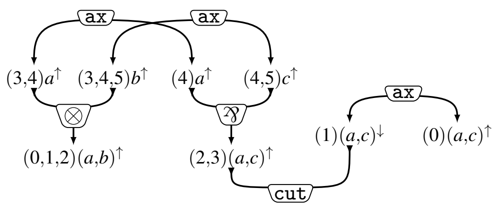
Figure 2: A MLL proof-structure R and an experiment of R

4 Semantic typing

We now describe graphically the main idea of our algorithm for MLL proof-structures, on two examples. In Figure 3, we try, starting from the conclusions, to build an experiment of R, one (Figure 3a) with putative result $((a,b),(a,b)) \in |(A \otimes B) \parr (A^\perp \parr B^\perp)|_\varnothing$, the other (Figure 3b) with putative result $((a,b),(a,c)) \in |(A \otimes B) \parr (A^\perp \parr B^\perp)|_\varnothing$ (with $b \neq c$). Tokens travel through R, encapsulating an element of the relational interpretation of the port they are sitting on. We depict the tokens at different steps (where each step is defined by one token moving). A token is depicted as its travel direction, its content, and the step on which the token exist: $(3,4)a^\uparrow$ meaning that a token is there on steps 3 and 4, containing a and going up.



(a) A succesfull typing



(b) A failing typing

Figure 3: Two typings or R

Let's describe the possible execution steps depicted in Figure 3a:

0. two upward tokens containing (a,b) are placed on the two free ports of R (the principal port of the $\otimes$-cell and the right principal port of the right axiom);

1. the right token goes up through the right axiom, and exits downwards from its left principal port;
2. the same token goes down through the cut and exits upwards from its left auxiliary port;
3. the other token (on the principal port of the $\otimes$-cell) gets split in two upwards tokens, one containing a on the left auxiliary port of the $\otimes$-cell, the other containing b on its right auxiliary port;
4. the right token containing (a,b) on the principal port of the $⅋$-cell gets split in two upwards tokens, one containing a, the other containing b;
5. the right token containing a goes up through an axiom, exits downwards from its other principal port and meets an upwards token containing a too. They annihilate each other;
6. the right token containing b goes up through an axiom, exits downwards from its other principal port and meets an upwards token containing b too. They annihilate each other.

As there are no more tokens on R, we say that the execution is successful, and so we proved the judgment $\rhd R : ((a,b),(a,b)) : (A \otimes B, A^\perp ⅋ B^\perp)$. Conversely, the same thing happens in the execution steps depicted in Figure 3b (replace the initial token (a,b) on the right with (a,c)), apart from the last step:

6. the right token containing b goes up through an axiom, exits downwards from its other principal port and meets an upwards token containing c. Nothing happens, the machine is stuck.

As the machine is stuck with tokens on it, we say that the execution has failed, and we proved the negation of the judgment $\rhd R : ((a,b),(a,c)) : (A \otimes B, A^\perp ⅋ B^\perp)$.

We will formalize this mechanism in Section 6.

We want our algorithm to be able to handle cases like the (non-cut-free) MLL proof-structure S in Figure 4, where part of the information carried by an experiment can not be retrieved from its conclusion.

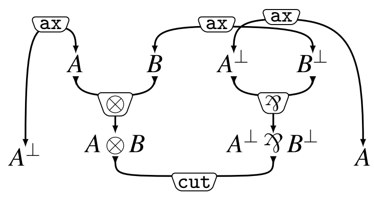


Figure 4: A cyclic proof-structure S

By simply propagating a relational point using the same strategy as defined before, we cannot guess which relational point should be the label of the port of type B through the experiment. In a way, all the information in the cycle concerning B in S (seen as an undirected graph) is hidden from the conclusions of S. We solve this problem by introducing variables (which we already forced into the definition of the relational web of a formula): some transitions can be fired with incomplete information which will be checked later.

5 Vector Addition Systems

Vector Addition Systems (VAS) (for instance, [15]) are a prominent class of infinite state systems. They comprise a finite number of counters ranging over natural numbers. When taking a transition, an integer can be added to a counter, provided the result stays non-negative. To give an example, let us consider a VAS with two counters. Let α be defined as the displacement $(1,-1)$. The transition $(1,2) \xrightarrow{\alpha} (2,1)$ is valid, while α does not define any transition starting from the state $(1,0)$, because $0+(-1)$ is negative.

VAS are particularly well-suited to represent systems with an infinite number of states. Our idea here is to encode the presence, direction and content of a token in a variant of VAS: with each port p is associated a counter, which is set to 0 if there is no token on p, it is set to a if there is an *upwards* token containing the relational point a on p, it is set to $-a$ if there is a *downwards* token containing a on p. While the systems studied in the sequel have an essentially finitary behaviour, it is not the case in extensions to the exponentials of LL: the relational type-checking of MELL proof-structures with exponentials will imply the presence of any arbitrary number of tokens on a given port.

All counters in VAS are natural numbers. We depart from traditional VASs for two reasons: we want to be able to encode tokens going up, but also going down. We also want counters to account for tokens containing a relational point, so the counters are to be taken is a space engendered by relational points with coefficients. For MLL proof-structures, we can restrict ourselves to the case where every coefficient is in $\mathbb{B}_3 = \{-1, 0, 1\}$. As such, the counter associated with a port $p \in \mathscr{P}(\Phi)$ is a formal series of points in $|\mathsf{tp}_\Phi(p)|_\mathscr{V}$ with coefficients in $\{-1, +1\}$: we denote by $\mathbb{B}_3[|\mathsf{tp}_\Phi(p)|_\mathscr{V}]$ the set of such formal series. It is endowed with a partial sum and a partial difference (by extending pointwise the partial sum and partial difference of $\mathbb{B}_3$, seen as a subset of $\mathbf{Z}$). The configuration of the machine ought then to be an element of the dependent product $\prod_{p \in \mathscr{P}(\Phi)} \mathbb{B}_3[|\mathsf{tp}_\Phi(p)|_\mathscr{V}]$, which we will write as a function.

6 The relational interaction abstract machine

We are now ready to give the formal definition of the Relational Interaction Abstract Machine for MLL, which decides semantic type judgments of the relational type-checking problem for MLL proof-structures.

Given a relation $R \subseteq A \times B$ and any $a \in A$ and $b \in B$, aRb stands for $(a,b) \in R$.

Definition 3 (Relational Interaction Abstract Machine for MLL)**.** Let Φ be a MLL proof-structure.

An *environment* is a finite partial map from $\mathscr{V}$ to $\mathbb{REL}$. The set of environments is denoted by $\mathfrak{Env}$.

The *relational interaction abstract machine* (RIAM) *for* MLL associated with Φ, denoted by M^Φ, has the following components:

- its alphabet is $\Sigma_\Phi = \mathfrak{Env} \cup \mathscr{C}(\Phi)$, where $\mathscr{C}(\Phi)$ is the set of cells of Φ (disjoint from $\mathfrak{Env}$);
- its set of configurations is $\prod_{p \in \mathscr{P}(\Phi)} \mathbb{B}_3[|\mathsf{tp}_\Phi(p)|_\mathscr{V}]$.

The transitions for the RIAM M^Φ are binary relations on the set of configurations of M^Φ labelled by an element of Σ_Φ. So, there are two kinds of transitions: the *displacement transitions* are labelled by a cell of Φ, while the *unification transitions* are labelled by an environment.

Formally, for any cell $c \in \mathscr{C}(\Phi)$, a displacement transition labelled by c is a binary relation $\xrightarrow{c}$ on the set of configurations of M^Φ defined by:

$$\text{for any configurations } x, x' \text{ of } M^\Phi, \quad x \xrightarrow{c} x' \;\text{ if }\; c\,\delta\,(x' - x),$$

where the *displacement relation* $\delta \subseteq \mathscr{C}(\Phi) \times \prod_{p \in \mathscr{P}(\Phi)} \mathbb{B}_3[|\mathsf{tp}_\Phi(p)|_\mathscr{V}]$ is the binary relation defined by:

- if c is of type ax, let $\mathsf{P}^{\mathsf{pri}}_\Phi(c) = \langle p, q \rangle$ and $\forall a \in |\mathsf{tp}_\Phi(p)|_\mathscr{V}$, $c\,\delta \begin{cases} p & \mapsto & -a \\ q & \mapsto & -a \\ r & \mapsto & 0, \text{if } r \neq p, q \end{cases}$
- if c is of type cut, let $\mathsf{P}^{\mathsf{aux}}_\Phi(c) = \langle p, q \rangle$ and $\forall a \in |\mathsf{tp}_\Phi(p)|_\mathscr{V}$, $c\,\delta \begin{cases} p & \mapsto & a \\ q & \mapsto & a \\ r & \mapsto & 0, \text{if } r \neq p, q \end{cases}$
- if c is of type 1 or $\bot$, let $\mathsf{P}^{\mathsf{pri}}_\Phi(c) = p$ and $c\,\delta \begin{cases} p & \mapsto & -() \\ r & \mapsto & 0, \text{if } r \neq p \end{cases}$
- if c is of type $\otimes$ or $⅋$, let $\mathsf{P}^{\mathsf{pri}}_\Phi(c) = q$ and $\mathsf{P}^{\mathsf{aux}}_\Phi = \langle p_1, p_2 \rangle$, and $c\,\delta \begin{cases} p_1 & \mapsto & \bullet \\ p_2 & \mapsto & \circ \\ q & \mapsto & -(\bullet, \circ) \\ r & \mapsto & 0, \text{if } r \neq p_1, p_2, q \end{cases}$.

  where $\bullet, \circ \in \mathscr{V}$ are two fresh variables.

For any environment $e \in \mathfrak{Env}$, an unification transition labelled by e is a binary relation $\xrightarrow{e}$ on the set of configurations of M^Φ defined by:

$$x \xrightarrow{e} e(x) \text{ if } \exists p \in \mathscr{P}(\Phi) \text{ such that } \begin{cases} x(p) = a_1 - a_2 + \vec{a},\ a_1 \neq a_2,\ \vec{a} \in \mathbb{B}_3[|\mathsf{tp}_\Phi(p)|_{\mathscr{V}}] \\ e = \text{m.g.u.}(a_1, a_2) \end{cases}$$

where m.g.u. denotes the most general unifier, and $e(x)$ denotes the configuration obtained from x by replacing each variable occurring in x and in the domain of the environment e with its image via e. Such a transition unifies (and so annihilates) two elements of a formal sum of opposite sign in one of the counters.

Given $\sigma = s_1 \cdots s_n \in \Sigma_\Phi^\star$ (*i.e.* σ is a word over the alphabet Σ_Φ) and two configurations x and x' of M^Φ, we define $x \xrightarrow{\sigma} x'$ by relational composition:

$$x \xrightarrow{\sigma} x' \text{ if } \exists (x_i)_{1 \leqslant i < n} \text{ such that } x \xrightarrow{s_1} x_1 \xrightarrow{s_2} \cdots \xrightarrow{s_{n-1}} x_{n-1} \xrightarrow{s_n} x'$$

(in particular, if $n = 0$ then $x = x'$). We then say that σ is an *execution* of M^Φ.

We denote by $\underset{\Phi}{\rightarrow}$ the *reachability binary relation* defined by: $x \underset{\Phi}{\rightarrow} x'$ if $\exists \sigma \in \Sigma_\Phi^\star$ such that $x \xrightarrow{\sigma} x'$.

Given a configuration $x \in \prod_{p \in \mathscr{P}(\Phi)} \mathbb{B}_3[|\mathsf{tp}_\Phi(p)|_{\mathscr{V}}]$ of M^Φ, we say that M^Φ *recognizes* x if $x \underset{\Phi}{\rightarrow} 0$, and that $\sigma \in \Sigma_\Phi^\star$ *recognizes* x if $x \xrightarrow{\sigma} 0$ (by abuse of notation, 0 is the configuration of M^Φ associating 0 with every port of Φ). We say then that σ is *successful* (on x).

Given a MLL proof-structure Φ of conclusion $\Gamma = (A_1, \dots, A_n)$ with n ordered free ports $p_1 < \cdots < p_n$, and given the point $\alpha = (\alpha_1, \dots, \alpha_n) \in |\invamp\Gamma|_{\mathscr{V}}$, we say that the RIAM M^Φ (or the word $\sigma \in \Sigma_\Phi^\star$) *recognizes* α if it recognizes the configuration of M^Φ associating 0 with every port of Φ which is not a free port, and α_i with p_i for any $1 \leq i \leq n$.

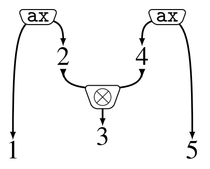


Figure 5: A named proof-structure Φ

As an example, we will describe the RIAM associated with the MLL proof-structure Φ in Figure 5. Its ports are numbered $1, 2, 3, 4, 5$, and its cells are named $\{\mathtt{ax}_{1,2}, \mathtt{ax}_{4,5}, \otimes\}$.

The RIAM M^Φ associated with Φ has $\Sigma_\Phi = \mathfrak{Env} \cup \{\mathtt{ax}_{1,2}, \mathtt{ax}_{4,5}, \otimes\}$ as alphabet, $|A|_{\mathscr{V}} \times |A|_{\mathscr{V}} \times |A \otimes B|_{\mathscr{V}} \times |B|_{\mathscr{V}} \times |B|_{\mathscr{V}}$ as set of configurations and its displacement relation δ is defined by (for $\circ$ and $\bullet$ fresh variables):

$$\begin{aligned} \forall a \in |A|_{\mathscr{V}},\ & \mathtt{ax}_{1,2}\ \delta\ (-a, -a, 0, 0, 0) \\ \forall b \in |B|_{\mathscr{V}},\ & \mathtt{ax}_{4,5}\ \delta\ (0, 0, 0, -a, -a) \\ & \otimes\ \delta\ (0, \circ, -(\circ, \bullet), \bullet, 0). \end{aligned}$$

The relational point $(a, (a, b), b)$ is recognized by M^Φ by the word $(\mathtt{ax}_{1,2}, \otimes, \{\bullet \mapsto b, \circ \mapsto a\}, \mathtt{ax}_{4,5})$.

7 Recognition of the relational interpretation

In this section we aim to prove that the RIAM provides an algorithm deciding the relational type-checking problem for MLL proof-structures in time linear in the size of the MLL proof-structures.

Among all executions of a RIAM, some are in normal form: intuitively, they do not create any tokens but only propagate those already present in the initial configuration.

Definition 4 (Normal execution)**.** Let Φ be a MLL proof-structure. An execution σ of M^Φ is *normal* if:

- for each displacement transition $x \xrightarrow{c} x'$ in σ, there is $p \in \mathscr{P}(\Phi)$ such that $x(p) \neq 0$ and $x'(p) = 0$;
- and each displacement transition $x \xrightarrow{c} x'$ in σ such that there exists $p \in \mathscr{P}(\Phi)$ such that $x'(p) = -a + a'$ is followed by an unification transition $x' \xrightarrow{e} x''$ such that $x''(p) = 0$.

Lemma 2. *Let Φ be a* MLL *proof-structure of conclusion Γ. Let $x \in |\parr\Gamma|_\varnothing$. A normal successful execution of M^Φ on x is at most of length $2\,\mathsf{card}(\mathscr{C}(\Phi))$, i.e. twice the number of cells of Φ.*

Lemma 3. *Let Φ be a* MLL *proof-structure with conclusion Γ. Let $x \in |\parr\Gamma|_\varnothing$ be such that M^Φ recognizes x. Then, there exists a normal execution of M^Φ that recognizes x.*

Normal executions of the RIAM can be used to define a partial experiment on all ports connected to the free ports, while an experiment can be sequentialized in an execution.

Theorem 4. *Let Φ be a* MLL *proof-structure of conclusion Γ. Let $x \in |\parr\Gamma|_\varnothing$.*
Then, $x \in [\![\Phi]\!]$ if and only if there exists a (normal) execution of M^Φ recognizing x.

As, moreover, an execution with a counter in a state of the form $a + b$, with a and b not unifiable, cannot be extended in a successful execution, we get:

Corollary 5. *The* RIAM *decides judgments of the form $\rhd\Phi : x : \Gamma$ in time $O(\mathsf{card}(\mathscr{C}(\Phi)))$ (i.e. in time linear in the size of Φ).*

8 Relationship with the Geometry of Interaction

The Geometry of Interaction (GoI) is a form of operational semantics proposed by Girard [8]. It can be presented as a compilation of terms into bi-deterministic automata [16, 1], via the token machine called Interaction Abstract Machine. We will only focus on multiplicative GoI and its relationship with the RIAM.

Let R be a MLL proof-structure, for simplicity without cells of type 1 and $\perp$. A *state* of the Interaction Abstract Machine (IAM) is given by a stack made of the symbols p and q, a port of R (showing the current position of the token) and a direction (upwards or downwards). Transitions of states are depicted graphically as follows (where, in the token, π is the stack, and $\cdot$ denotes the cons operation):

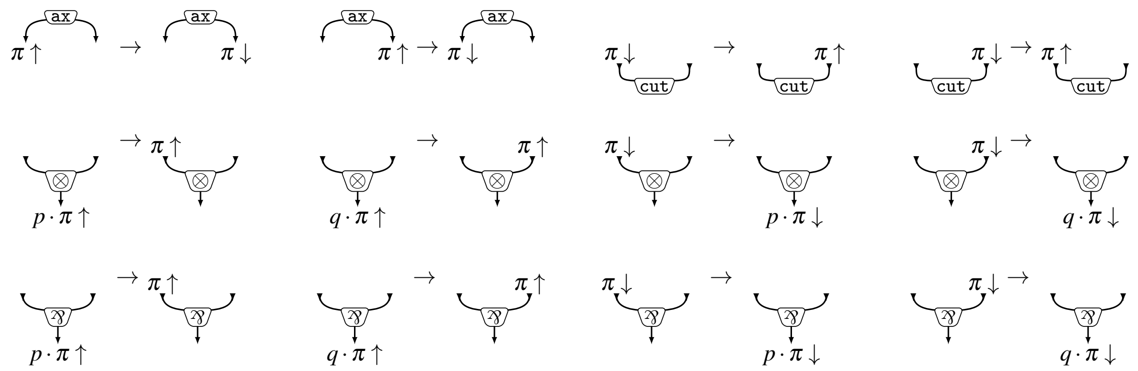


Figure 6: Transitions of the IAM

Remark that all transitions are reversible, that is, a transition taken in reverse (with tokens going in the opposite direction) is still a transition. If σ is a sequence of transitions, we denote by $\sigma^\star$ the sequence of transitions in reverse, with directions reverted.

A *successful run* $(r, \pi) \to^* (r', \pi')$ of the IAM is a finite sequence of transitions starting on the free port r of R with stack π and ending on the free port r' of R with stack π'.

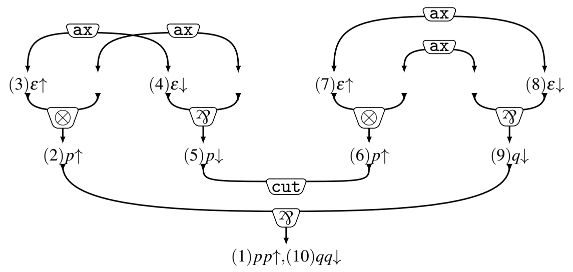


Figure 7: A run of the IAM starting on the stack pp

Example 1. Figure 7 represents a run of the IAM starting on the stack pp. It ends on the stack qq. By reversibility, the run starting on the stack qq also stops on the stack pp.

To state the relationship between IAM and RIAM, we need the following notion. Given $x \in \mathbb{REL}$ and $a \in \mathscr{A}t$, the *projection* $x_{|a}$ *of* x *on* a is a set of stacks defined recursively as (ε denotes the empty stack):

$$a_{|a} = \{\varepsilon\}, \quad b_{|a} = \varnothing \text{ for any } b \in \mathscr{A}t \text{ with } b \neq a, \quad (x_1,x_2)_{|a} = \{p \cdot \pi_1 \mid \pi_1 \in x_{1|a}\} \cup \{q \cdot \pi_2 \mid \pi_2 \in x_{2|a}\}.$$

Theorem 6. *Let R be a* MLL *proof-structure without cells of type* 1 *and* $\bot$, *let r,r' be two free ports of R.*

Let s be a successful run $(r,\pi) \to^* (r',\pi')$ of the IAM. *Let $x \in [\![R]\!]$. For every partition of s in two sequences (each containing at least one transition) s_1 and s_2, there exist two executions σ_1, σ_2 of the* RIAM M^R *such that:*

- *σ_1 is of the same length as s_1, σ_2 is of the same length as s_2;*
- *before the n-th transition of σ_1, there is a relational token in the port where the token is before the n-th transition of s_1, that is moved by the said transition;*
- *before the n-th transition of σ_2, there is a relational token in the port where the token is before the n-th transition of $s_2^\star$, that is moved by the said transition;*
- *let l be the port where the token of s_1 and $s_2^\star$ ends, the sequence $\sigma_1\sigma_2$ of transitions of the* RIAM *starting on w, either ends with no token on l or can be followed by a unification transition that empties l.*
- *$\sigma_1\sigma_2$ can be extended to a successful execution on x.*

Conversely, let $x \in [\![R]\!]$ be an injective point (meaning that every atom appearing in x appears exactly twice) and let e *be an experiment such that $|\mathsf{e}| = x$. For every atom a appearing in x, there are two stacks π_1 and π_2 such that $x_{|a} = \{\pi_1, \pi_2\}$, and if r and r' are the two free ports of R such that the atom a appears in $\mathsf{e}(r)$ and $\mathsf{e}(r')$, then there is a successful run $(r,\pi_1) \to^* (r',\pi_2)$ of the* IAM.

Theorem 6 justifies our slogan that the multiplicative GoI is a partial relational type-checking.

Example 2. The run of the IAM in Figure 7 split (for instance) in the two sequences of transitions (1)–(8) and (8)–(10) induces an execution of the RIAM that can be extended on a successful execution on the relational point $((a,b),(b,a))$, by Theorem 6. The execution (as depicted in Figure 8) ends with four tokens still on the MLL proof-structure, that can be brought together and annihilated.

Reciprocally, the point $((a,b)(b,a))$ projected on a (resp. b) induces the stacks pp and qq (resp. pq and qp) and indeed, the run of the IAM starting on pp (resp. pq) returns qq (resp. qp).

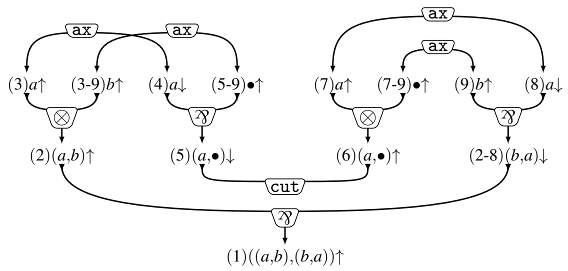


Figure 8: A RIAM execution induced by the IAM run in Figure 7

9 Extending to the lambda-calculus (sketch)

The techniques showed in this article for MLL can be extended to MELL and hence to the (simply typed or untyped) λ-calculus, thanks to Girard's encoding of λ-calculus into MELL proof-structures through the call-by-name translation of the intuitionistic arrow $\alpha \to \beta$ into $!\alpha \multimap \beta$ (and the recursive types identity $o = o \to o$, *i.e.* $o = !o \multimap o$, in the untyped case). The connectives of MELL are those of MLL to which are added two new unary connectives, the exponentials ! and ?. The syntax of MELL proof-structures is enriched with a box constructor, taking a MELL proof-structure and encapsulating it in a box in order to mark a resource that can be duplicated or erased under the cut-elimination process.

The interpretation of the exponential cells (*i.e.* the notion of experiment for MELL proof-structures) is a bit tricky to define. An elegant way is described in [12]. It is multiset based: the interpretation of a formula $!A$ or $?A$ is a multiset. The interpretation of a box is the multiset containing multiple interpretations of the content of the box.

We extend the definition of the RIAM in order to handle the exponentials:

- coefficients of the machine have to be taken in $\mathbf{Z}$ (and no more in $\mathbb{B}_3$), moreover on top of containing a relational point, tokens contain also a stack of timestamps remembering when boxes are entered;
- new rules must be added, allowing to pass through exponential cells. They amount to splitting the content of multi-cells in all possible ways;

Together with a suitable definition of normal execution, and using techniques similar to [18], this allows to prove (where $|M|$ is the size of the λ-term M and $|x|$ is the number of atoms occurring in $x \in \mathbb{REL}$):

Theorem 7. *The* RIAM *for the λ-calculus decides judgments of the form $\rhd M : x : \Gamma$ in time $O(|M| \times |x|)$.*

The proof of Theorem 7 is based on the property that the MELL proof-structures translating λ-terms enjoy some kind of connectedness: this property does no longer hold in the full MELL framework in general. To find an analogous of Theorem 7 for the full MELL framework is still an open question.

As a corollary of Theorem 7 and Theorem 1, we get the following (unpublished) result:

Corollary 8 (Terui, 2012)**.** *Let $\mathbf{W} := (o \to o) \to (o \to o) \to o \to o$ be the Church encoding of binary strings. Let M be a closed λ-term of type $\mathbf{W} \to \mathbf{B}$. It decides a language $\mathscr{L}$. Moreover, $\mathscr{L}$ is in* **LinTIME** *(deterministic linear time).*

The proof consisting of checking, for an encoded string $s : \mathbf{W}$, whether $\rhd Ms : [*] \to [\,] \to * : \mathbf{B}$, which is done in time linear in the size of the translation of M, itself linear in the size of M.

The result is surprising, as simply-typed λ-terms of type $\mathbf{N} \to \mathbf{N}$ (where $\mathbf{N}$ is the Church encoding of natural numbers) can represent a function of complexity an arbitrary tower of exponentials.

Acknowledgements Work partially supported by ANR projects COQUAS ANR-12-JS02-006-01 and ELICA ANR-14-CE25-0005. The authors thank Damiano Mazza for pointing to them the last Corollary.

References

[1] Vincent Danos and Laurent Regnier. Reversible, Irreversible and Optimal -machines. *Theoretical Computer Science*, 227(1-2):79–97, 1999.

[2] Daniel de Carvalho. Execution time of λ-terms via denotational semantics and intersection types. To appear in *Mathematical Structures in Computer Science*, 2009. Available at `http://arxiv.org/abs/0905.4251`.

[3] Daniel de Carvalho. The relational model is injective for Multiplicative Exponential Linear Logic. To appear in the proceedings of *CSL 2106*, 2016. Available at `http://arxiv.org/abs/1502.02404`.

[4] Daniel de Carvalho and Lorenzo Tortora de Falco. A semantic account of strong normalization in linear logic. *Inf. Comput.*, 248:104–129, 2016.

[5] Daniel de Carvalho, Michele Pagani, and Lorenzo Tortora de Falco. A semantic measure of the execution time in Linear Logic. *Theoretical Computer Science, Special issue Girard's Festschrift*, 412(20):1884–1902, 2011.

[6] 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, September 2012.

[7] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[8] Jean-Yves Girard. Towards a Geometry of Interaction. *Contemporary Mathematics*, 92:69–108, 1989.

[9] Stéphane Graham-Lengrand and Alexis Bernadet. Non-idempotent intersection types and strong normalisation. *Logical Methods in Computer Science*, Volume 9, Issue 4(4), October 2013.

[10] Charles Grellois and Paul-André Melliès. Relational Semantics of Linear Logic and Higher-order Model Checking. In *24th EACSL Annual Conference on Computer Science Logic, CSL 2015*, pages 260–276, 2015.

[11] 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*, pages 20:1–20:18, 2016.

[12] Giulio Guerrieri, Luc Pellissier, and Lorenzo Tortora de Falco. Syntax, Taylor expansion and relational semantics of MELL proof-structures: an unusual approach. Technical report, 2016. Available at `http://logica.uniroma3.it/~tortora/mell.pdf`.

[13] André Joyal, Ross Street, and Dominic Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(03):447–468, April 1996.

[14] Olivier Laurent. A Token Machine for Full Geometry of Interaction (Extended Abstract). In *Typed Lambda Calculi and Applications (TLCA'01)*, volume 2044 of *Lecture Notes in Computer Science*, pages 283–297. Springer, May 2001.

[15] Jérôme Leroux. The general vector addition system reachability problem by Presburger inductive invariants. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS 2009*, pages 4–13, 2009.

[16] Ian Mackie. The Geometry of Interaction Machine. In *Proceedings of the 22Nd ACM SIGPLAN Symposium on Principles of Programming Languages*, POPL '95, pages 198–208, New York, NY, USA, 1995. ACM.

[17] Luca Paolini, Mauro Piccolo, and Simona Ronchi Della Rocca. Essential and relational models. *Mathematical Structures in Computer Science*, FirstView:1–25, September 2015.

[18] Kazushige Terui. Semantic Evaluation, Intersection Types and Complexity of Simply Typed Lambda Calculus. In *23rd International Conference on Rewriting Techniques and Applications, RTA 2012*, pages 323–338, 2012.

[19] Lorenzo Tortora de Falco. Obsessional Experiments For Linear Logic Proof-Nets. *Mathematical Structures in Computer Science*, 13(6):799–855, December 2003.

A Appendix: Proof of the Main Theorem

Lemma 9. *Let Φ be a connected* MLL *proof-structure of conclusion Γ. Let $x \in |\parr\Gamma|_{\varnothing}$ be such that there exists a normal execution $\sigma \in \Sigma^\star$ of M^Φ recognizing x.*

It defines an experiment e^x_σ of Φ such that $|\mathsf{e}^x_\sigma| = x$.

This Lemma follows directly from the following:

Lemma 10. *Let Φ be a connected* MLL *proof-structure.*

Let x be a relational point such that there exists a normal execution $\sigma \in \Sigma^\star$ of M^Φ recognizing x.

σ is of the form $\sigma_1 \cdots \sigma_m$, where each σ_i is a sequence of displacement transitions followed by a unification transition (except eventually σ_m).

Every prefix $\sigma_1 \cdots \sigma_i$ of σ defines a partial experiment of Φ such that e is defined on exactly all ports whose counter has been non-null during the execution of the prefix.

Proof. If $m = 0$, then $\bar{x} = 0$, and there is no such $x \in \mathbb{REL}$. Moreover, Φ has no conclusion. The relational interpretation of a proof-structure with no conclusion is empty, which is coherent.

Suppose $m \geqslant 1$. By induction, let $0 \leqslant i < m$. The empty word is a prefix of σ and defines a partial experiment of Φ. Suppose $\sigma_1 \cdots \sigma_i$ defines a partial experiment e of Φ. σ_{i+1} is a succession.

σ_{m+1} starts with an element of $\mathscr{C}(\Phi) \times \{\mathsf{d}, \mathsf{l}, \mathsf{r}\}$, either e is defined on all ports of c, and e still verifies the property for $\sigma_1 \cdots \sigma_{m+1}$, or e is not defined on one of the ports of c. In this case, σ_{m+1} adds or substracts a relational element a to every such port p. If σ_{m+1} contains no substitution, we pose $\mathsf{e}(p) = a$ and check that it is a partial experiment. If σ_{m+1} contains a substitution s, we pose $\mathsf{e}(p) = a$ and verify that $s \circ \mathsf{e}$ is a partial experiment.

By induction, we proved that any prefix of σ defines a partial experiment of Φ. ☐

Proof of Lemma 9. As Φ is connected, the partial experiment defined by σ is defined on every ports of Φ. It defines an experiment of Φ by chosing interpretations of the variables. ☐

Lemma 11. *Let Φ be a connected* MLL *proof-structure and e be an experiment of Φ.*

There exists a normal execution $\sigma \in \Sigma^\star$ of the machine M^Φ recognizing the result of e, such that $\mathsf{e} = \mathsf{e}^{|\mathsf{e}|}_\sigma$.

Proof. The cells of Φ are naturally equipped with a partial order: the one represented bottom-up in the graphical representation. It can be refined in a total order. The execution starting from x firing every transition in order followed greedily by unification transition is normal and recognizes x. ☐

From which we deduce:

Proof of Theorem 4. Φ can be decomposed in connected components, some of which have no conclusions (and play no role, both in term of relational interpretation than in term of execution) and some which are connected with conclusions. The two Lemma 9 and Lemma 11 apply, which concludes. ☐