Relational type-checking of connected proof-structures

Giulio Guerrieri\textsuperscript{1,2}, Luc Pellissier\textsuperscript{3}, and Lorenzo Tortora de Falco\textsuperscript{1}

\textsuperscript{1} Dipartimento di Matematica e Fisica, Università Roma Tre Rome, Italy \{gguerrieri,tortora\}@uniroma3.it
\textsuperscript{2} Institut de Mathématiques de Marseille, UMR 7373, Aix-Marseille Université F-13453 Marseille, France
\textsuperscript{3} LIPN, UMR 7030, Université Paris 13, Sorbonne Paris Cité F-93430 Villetaneuse, France luc.pellissier@lipn.univ-paris13.fr

Abstract

It is possible to define a typing system for Multiplicative Exponential Linear Logic (MELL): in such a system, typing judgments are of the form \( \vdash R : x : \forall \Gamma \), where \( R \) is a MELL proof-structure, \( \Gamma \) is the list of types of the conclusions of \( R \), and \( x \) an element of the relational interpretation of \( \forall \Gamma \), meaning that \( x \) is an element of the relational interpretation of \( R \) (of type \( \forall \Gamma \)).

As relational semantics can be used to infer execution properties of the proof-structure, these judgment can be considered as forms of quantitative typing.

We provide an abstract machine that decides, if \( R \) satisfies a geometric condition, whether the judgment \( \vdash R : x : \forall \Gamma \) is valid. Also, the machine halts in bilinear time in the sizes of \( R \) and \( x \).

1 Introduction

Intersection types have been introduced as a way of extending the \( \lambda \)-calculus’ simple types with finite polymorphism. This is done by adding a new type constructor \( \cap \) and new typing rules. A term of type \( A \cap B \) can be used in an ulterior derivation both as data of type \( A \) or of type \( B \). Contrarily to simple types (which are sound but incomplete), intersection types are a sound and complete characterization of strong normalization.

Intersection types were originally idempotent, that is, the equation \( A \cap A = A \) held. This corresponds to an interpretation of a type typed as \( M : A \cap B \) as \( M \) can be used as data of type \( A \) or as data of type \( B \). In a non-idempotent setting (i.e. by dropping the equation \( A \cap A = A \)), the meaning of the typing judgment is strengthened in \( M \) can be used once as data of type \( A \) and once as data of type \( B \). Non-idempotent intersection types have been used to get qualitative and quantitative information on the execution time of \( \lambda \)-terms [1, 5].

Relational semantics is one of the simplest semantics of \( \lambda \)-calculus (and linear logic). A type is interpreted by a set, and a \( \lambda \)-term (or linear logic proof-structure) by a relation between sets which is invariant under \( \beta \)-reduction (and cut-elimination). It happens that the relational semantics corresponds to a non-idempotent intersection types system, called System \( R \) in [1] (see also [7]): a type derivation of a \( \lambda \)-term in System \( R \) corresponds to an experiment (see [4]) of a linear logic proof-structure, and the conclusion of such a type derivation corresponds to the result of this experiment i.e. a point in the relational semantics. So, knowing that an element is or not in the relational interpretation of a \( \lambda \)-term (or linear logic proof-structure) already gives a lot of information on the execution of this \( \lambda \)-term (or linear logic proof-structure) [1, 2]. For instance, given two correct (i.e. arising from a derivation on the sequent calculus) MELL proof-structure \( \pi_1 \) and \( \pi_2 \) without cuts, it is
possible to compute whether $\pi_1$ and $\pi_2$ can be composed and the length of the reduction to the normal form of this composition.

We introduce semantical typing judgments of the form $\vdash R : x : \emptyset$, where $R$ is a MELL (the multiplicative-exponential fragment of linear logic) proof-structure whose conclusion is the list of MELL formulæ $\emptyset$, and $x$ in the interpretation in the relational model of the MELL formula $\emptyset$. Our goal is to decide in a tractable way whether a judgment of this form is valid or not, i.e. whether $x$ is a point of the relational semantics of $R$ or not.

We thus define the Relational Interaction Abstract Machine (Section 5) able to decide such judgments on a fragment of all MELL proof-structures, that works by moving tokens embodying relational elements through the proof-structure. The machine moreover stops on a sequent $\vdash R : x : \emptyset$ after a number of steps bilinear in the size of $x$ and of $R$.

The class of MELL proof-structures on which our machine is sound and complete, defined in Section 4, is moreover quite natural and large enough to contain the $\lambda$-calculus.

As a corollary, we prove that languages decided by simply-typed $\lambda$-terms of type $\text{Str}[A/X] \to \text{Bool}$ are in $\text{LinTIME}$ (deterministic linear time).

## 2 Elements of MELL syntax

We set $\mathcal{L}_{\text{MELL}} = \{1, \bot, \otimes, \exists, !, ?, ax, cut\}$. The MELL connectives are $1, \bot, \otimes, \exists, !, ?$. We say that $1, \bot, \otimes, \exists$ (resp. $!, ?$) are the multiplicative (resp. exponential) connectives, and $1, \bot$ are the units.

The set of MELL formulas is generated by the grammar:

$$A, B, C ::= X | X \bot | 1 | \bot | \otimes | A \otimes B | A \exists B | !A | ?A.$$  

where $X$ ranges over a infinite countable set of propositional variables.

Proof-structures offer a syntax for MELL proofs. They are direct labelled graphs $\Phi$ built from the cells: We call ports the wires of such graphs, divided in principal ports (depicted down in the picture) and auxiliary ports (depicted up).

They are moreover endowed with a function $\boxempty_\Phi$ from the $?$ and cut cells to auxiliary ports of ! cells such that:

- all cells in the image of $\boxempty_\Phi$ have exactly one auxiliary port;
- inclusion of proof-structures obtained by choosing all cells which are above cells of the same image through $\boxempty_\Phi$ is a tree-like order.

We say that a proof-structure is a MELL proof-structure if all !-cells are in the image of $\boxempty_\Phi$. We say that it is a DiLL$_0$ proof-structure if $\boxempty_\Phi$ is the empty function.
Elements of relational semantics

We define relational experiments straightforwardly on \( \text{DiLL}_0 \) proof-structures (that is, proof-structures without boxes, but with arbitrary co-structural cells) by adapting the definition in [4]: a partial experiment is a function associating with a link an element of the interpretation of its type coherently with the structure. We define the relational semantics of a \( \text{DiLL}_0 \) proof-structure as the set of results of its experiments, i.e. the image of its conclusions through its experiments. The relational semantics of a \( \text{MELL} \) proof-structure \( R \) is just the union of the relational semantics of the \( \text{DiLL}_0 \) proof-structures in the Taylor expansion of \( R \) [3].

The Taylor expansion acts as a bridge between syntax and semantics, allowing to retain the simplicity of the multiplicative fragment while expanding it to the full \( \text{MELL} \).


3-connection

We now introduce the fragment on which our algorithm will act: 3-connected \( \text{MELL} \) proof-structures.

**Definition 1 (3-path, 3-accessibility).** Let \( R \) be a \( \text{MELL} \) proof-structure.

A 3-path on \( R \) (from \( p_0 \) to \( p_n \)) is a finite sequence \((p_0, \ldots, p_n)\) of ports of \( R \) obtained by applying a finite number of times the following rules:

1. \((p)\) is a 3-path for any \( p \) port of \( R \);
2. if \( \vec{p} = (p_0, \ldots, p_n) \) is a 3-path where \( p_n \) is a port of a cell \( l \) of \( R \) of type not 3, then \( \vec{p} \cdot q \) is a 3-path, for any \( q \) port of \( l \);
3. if \( \vec{p} = (p_0, \ldots, p_n) \) is a 3-path where \( p_n \neq p_0 \) is a port of a 3-cell \( l \) of \( R \), and if for all ports \( r \) of \( l \), save at most one, there is a 3-path from \( p_0 \) to \( r \), then \( \vec{p} \cdot q \) is a 3-path, for any \( q \) port of \( l \).

For every port \( p \) of \( R \), the set of the 3-accessible ports from \( p \) in \( R \) is

\[ \text{access}_R^3(p) = \{ q \in P_R \mid \text{there is a 3-path in } R \text{ from } p \text{ to } q \} \].

**Definition 2 (3-path inside a box, 3-connectedness).** Let \( R \) be a \( \text{MELL} \) proof-structure.

Given a !-cell \( l \), a 3-path \( \vec{p} = (p_0, \ldots, p_n) \) in \( R \) is inside the box of \( l \) if \( p_i \) is in the box of \( l \) for any \( 0 \leq i \leq n \).

\( R \) is 3-connected if
Relational type-checking of connected proof-structures

for any l-cell l and any port p inside the box of l, there is a \( ? \)-path inside the box of l from the principal door of the box of l to p;
all the ports at depth 0 are \( ? \)-accessible from the conclusions.

This technical condition arises from the algorithm presented next. Nonetheless, the fragment of \( ? \)-connected proof-structures is quite general: all MELL proof-structures which are translations of \( \lambda \)-terms are \( ? \)-connected.

Recognition of the relational interpretation

We now introduce the main object of this article: the Relational Interaction Abstract Machine that decides the semantic sequents. The notation is inspired by Danos, Régnier, Mackie and Laurent’s Interaction Abstract Machine [6]. Indeed, this work has a distinct Geometry of Interaction flavour.

The definition of the machine is in the Appendix. The main idea behind it is that its state is composed of tokens containing a relational element that travel through the proof-structure, obeying type-directed rules. For instance, whenever a token goes up through a \( \otimes \) cell, it splits into two tokens, one going left, one going right. A token going up and one going down containing the same relational element anihilate when they meet. The only thing that could cause non-determinism are the contractions, where we don’t know how to split a multiset between the different branches: that’s why the \( ? \)-connection condition restricts the way contractions arise in the structures.

Lemma 3. Let \( R \) be a MELL proof-structure whose conclusions are ordered.
A successful run of \( M^R \) defines a partial experiment of \( R \).
Reciprocally, an experiment of \( R \) defines a successful run of \( M^R \).

Theorem 4. Let \( R \) be a \( ? \)-connected MELL proof-structure whose conclusions \( \Gamma \) are ordered, and let \( x \in [\forall \Gamma] \).
The point \( x \) is in the relational interpretation of \( R \) iff \( M^R \) runs successfully on \( x \).
Moreover, if \( R \) is acyclical, if we write \( |x| \) the number of atoms appearing in \( x \) and \( \text{size}(R) \) the number of links in \( R \), the machine halts after \( O(|x| \times \text{size}(R)) \).
The Relational Interaction Abstract Machine decides sequents of the form \( \vdash R : x : \forall \Gamma \), when \( R \) is acyclical and \( ? \)-connected, in bilinear time in the sizes of \( R \) and \( x \).
In particular, the machine decides in bilinear times such sequents for correct proof-structures.

This result can be used in the following special case: we know (from the aforelinked long version) that a certain point (an injective 2-point) of a \( ? \)-connected MELL proof-structure characterizes entirely the proof-structure. So our algorithm can answer the following question: given a \( ? \)-connected MELL proof-structure \( R \) (of conclusions \( \Gamma \)) and a cut-free MELL proof-structure \( S \) (with the same conclusions), is \( S \) the normal form of \( R \)?
In the general case, there is no better algorithm than performing the cut-elimination on \( R \) and verifying whether the resulting proof-structure is isomorphic to \( S \). In the box-connected case, it suffices to compute an injective 2-point of \( S \) (which faithfully represents \( S \)) and to verify that it is an element of the interpretation of \( R \).

Definition 5 (Injective 2-point). An injective 2-point is a point \( x \) of the relational interpretation of a MELL-proof structure \( R \) such that:
each atom appearing in \( x \) appears exactly twice;
Every multiset in $x$ corresponding to a co-contraction in the difnet from which it arose is of cardinality (counted with its multiplicity) 2.

Every MELL proof-structure has injective 2-points. They are moreover all equivalent under the substitution of atoms.

Theorem 6. If $R$ and $S$ are two ?-connected MELL proof-structures of same (ordered) conclusions, and $S$ is moreover cut-free, $M^R$ runs successfully on any 2-point of $S$ if and only if $S$ is isomorphic to the normal form of $R$.

We use here ?-connection twice: the recognition algorithm requires $R$ to be ?-connected, and ?-connection allows us to limit ourselves to having to check the 2-point of $S$.

The main theorem also have an interesting corollary, proven but unpublished by Terui:

Theorem 7 (Terui, 2012). Let

$$\text{Str} := !((X \rightarrow X) \rightarrow !((X \rightarrow X) \rightarrow X) \rightarrow X \rightarrow X)$$

$$\text{Bool} := !X \rightarrow !X \rightarrow X$$

be the linear-logic translations of Church binary strings and booleans.

Let $R$ be a simply-typed MELL-proof structure of type $\text{Str[A/X]} \rightarrow \text{Bool}$, for arbitrary $A$. It decides a language $L$.

If $R$ is ?-connected, then $L$ is in LinTIME (deterministic linear time).

The result is surprising, as ?-connected proof-structures encompass the call-by-name translation of simply-typed $\lambda$-calculus, and simply-typed $A$-terms of type $\text{Nat[A/X]} \rightarrow \text{Nat}$ can represent a function of complexity an arbitrary tower of exponentials.

Acknowledgements

The authors thank Damiano Mazza for making them aware of Terui’s theorem. Work partially supported by ANR projects COQUAS ANR-12-JS02-006-01 and ELICA ANR-14-CE25-0005.

References


A The Machine: Formally

Definition 8 (Relational Interaction Abstract Machine). Let \( R \) be a MELL proof-structure where \( \Gamma \) is the list of its (ordered) conclusions.

A state of the machine \( M^R \) associated with the MELL proof-structure \( R \) is a multiset of tokens \( A^\uparrow(p, x, s) \) where

- \( A \) is a MELL formula,
- \( \uparrow \in \{\uparrow, \downarrow\} \),
- \( p \) is a port of \( R \) of type \( A \),
- \( x \in |A| \) or \( x = 0 \), with 0 neutral for multiset sum,
- \( s \) is a stack of box-cells of \( R \).

The machine follows the transitions of Figures 3, 4 and 5: a rule of the form \( \frac{P'}{P} \) removes from the state the tokens on the left and adds to it the tokens on the right, if the guard condition \( P \) and \( P' \) are verified. Notations of the Figures:

- \( c \) (respectively \( d \)) is always the cell of \( R \) such that \( p \) is its principal (respectively auxiliary) port, when it exists and is unique;
- \( \mathbf{P}^\text{pri}_R(c) \) denotes the principal ports of \( c \); it is either a set (denoted by curly brackets \( \{\cdot\} \)) or an ordered pair (denoted by angle brackets \( \langle\cdot\rangle \));
- \( \mathbf{P}^\text{aux}_R(c) \) denotes the auxiliary ports of \( c \); it is either a set (denoted by curly brackets \( \{\cdot\} \)) or an ordered pair (denoted by angle brackets \( \langle\cdot\rangle \));
- \( \text{auxd}_R(l) \) denotes the auxiliary doors of a box rooted in the cell \( l \);
- \( \text{tp} \) is the type of a port.

A run of \( M^R \) on \((x_1, \ldots, x_n) \in \Gamma\) is any succession of transitions with the machine initialized in the state

\[
A^\uparrow \left( \text{concl}_R(i), x_i, \varepsilon \right), 1 \leq i \leq n
\]

where \( \text{concl}_R(i) \) is an enumeration of the conclusions of \( R \).

We say that \( M^R \) accepts \((x_1, \ldots, x_n) \) if there exists a run of \( M^R \) on \((x_1, \ldots, x_n) \) that halts on the empty state.
\[
\begin{align*}
A^\uparrow (p, a, s) & \xrightarrow{\text{cut}} A^\uparrow (p', a, s) \\
A^\downarrow (p, a, s) & \xrightarrow{\Phi} A^\uparrow (p', a, s) \\
A^\downarrow (p, a, s) & \xrightarrow{\emptyset} \emptyset \\
A \otimes B^\uparrow (p, (a, b), s) & \xrightarrow{\emptyset} A^\uparrow (p, a, s) \rightarrow B^\uparrow (p, b, s) \\
A \otimes B^\uparrow (p, (a, b), s) & \xrightarrow{\emptyset} A^\uparrow (p, a, s) \rightarrow B^\uparrow (p, b, s) \\
A^\uparrow (p, a, s) \rightarrow B^\uparrow (p, b, s) & \xrightarrow{\emptyset} A \otimes B^\uparrow (p, (a, b), s) \\
A^\uparrow (p, a, s) \rightarrow B^\uparrow (p, b, s) & \xrightarrow{\emptyset} A \otimes B^\uparrow (p, (a, b), s)
\end{align*}
\]

**Figure 3** Multiplicative transitions

\[
\begin{align*}
A^\uparrow (p, a, s) & \xrightarrow{\emptyset} A \otimes B^\uparrow (p, (a, \bullet), s) \rightarrow B^\uparrow (p, b, s) \\
B^\uparrow (p, b, s) & \xrightarrow{\emptyset} A \otimes B^\uparrow (p, (a, \bullet), s) \rightarrow A^\uparrow (p, \bullet, s) \\
A^\uparrow (p, a, s) & \xrightarrow{\emptyset} A \otimes B^\uparrow (p, (a, \bullet), s) \rightarrow B^\uparrow (p, \bullet, s) \\
B^\uparrow (p, b, s) & \xrightarrow{\emptyset} A \otimes B^\uparrow (p, (a, \bullet), s) \rightarrow A^\uparrow (p, \bullet, s)
\end{align*}
\]

**Figure 4** Cyclicity transitions. \(\bullet\) denotes a fresh variable.
\[
A^!(p, [a_1, \ldots, a_n], s) \xrightarrow{c!} A^!(p', a_1, s \cdot p), \ldots, A^!(p', a_n, s \cdot p)
\]

\[
A^!(p, [], s) \xrightarrow{c!} \text{tp}(p_1)^!(p_1, 0, s), \ldots, \text{tp}(p_n)^!(p_n, 0, s)
\]

\[
?A^!(p, [], s) \xrightarrow{c?} \emptyset
\]

\[
A^!(p_1, a_1^1, s \cdot l_1^1 \cdots l_{k_1}^1)
\]

\[
\ldots
\]

\[
A^!(p_n, a_n^m_1, s \cdot l_n^1 \cdots l_{k_n}^1)
\]

\[
A^!(p_1, a_1^1, s \cdot l_1^1 \cdots l_{k_1}^1)
\]

\[
\ldots
\]

\[
A^!(p_n, a_n^m_1, s \cdot l_n^1 \cdots l_{k_n}^1)
\]

\[
A^!(p_1, a_1^1, s \cdot l_1^1 \cdots l_{k_1}^1)
\]

\[
\ldots
\]

\[
A^!(p_n, a_n^m_1, s \cdot l_n^1 \cdots l_{k_n}^1)
\]

\[
?A^!(p, [a_1^1, \ldots, a_n^m_1, a_1', \ldots, a_p'], s)
\]

\[
\text{Figure 5} \quad \text{Exponential transitions.}
\]