Linear Realisability Over Nets and Second Order Quantification (short paper)

Adrien Ragot[1,2,*,†], Thomas Seiller[1,3,‡] and Lorenzo Tortora de Falco[2]

[1]*Université Sorbonne Paris Nord, France – LIPN UMR 7030*

[2]*Università Degli Studi Roma Tre, Italy – Dipartimento di Matematica e Fisica*

[3]*CNRS, France*

Abstract

We present a new realisability model based on othogonality for Linear Logic in the context of nets – untyped proof structures with generalized axiom. We show that it adequately models second order multiplicative linear logic.

As usual, not all realizers are representations of a proof, but we identify specific types (sets of nets closed under bi-othogonality) that capture exactly the proofs of a given sequent. Furthermore these types are orthogonal's of finite sets; this ensures the existence of a correctnesss criterion that runs in finite time.

In particular, in the well known case of multiplicative linear logic, the types capturing the proofs are generated by the tests of Danos-Regnier, we provide - to our knowledge - the first proof of the folklore result which states "test of a formula are proofs of its negation".

Keywords

Linear Logic, Realisability, Second Order Quantification, Proof Nets, Orthogonality, Correctness Criterion

Context

Realisability is a technique that extracts the computational content of proofs [1]. It was first introduced in 1945 by Kleene for Heyting Arithmetic – an Intuitionnistic axiomatization of arithmetic – based on the codes of Gödel's partial recursive functions [2]. Fixing an untyped computational model, the methodology of Realisability is based on two aspects:

- Types are given a computational status: the interpretation of a type[1] A is a set of programs $[\![A]\!]$, which behave similarly – its element are called *realizers* of A.
- A simple process transforming the proofs of the realized logic in programs is defined, introducing a non trivial predicate on programs, namely, some programs represent a proof, the *correct* programs, while others do not, the *incorrect* programs.

ICTCS 2023: 24th Italian Conference on Theoretical Computer Science, Palermo, Italy 13-15 September, 2023.

*Corresponding author.

†The author is supported by a VINCI PhD fellowship from the Franco-Italian Université.

‡The author is partially supported by the DIM RFSI project CoHOp, and the ANR ANR-22-CE48-0003-01 project DySCo.

ragot@lipn.fr (A. Ragot); seiller@lipn.fr (T. Seiller); tortora@uniroma3.it (L. Tortora de Falco)

[1]Equivalently, having the Curry–Howard correspondence in mind, A is a formula.

> For instance, whenever the computational model is a freely–generated language equipped with a binary relation capturing program execution, correct programs correspond to well–formed terms.

A theorem of *adequacy* or *soundness* usually follows, e.g. each proof of A corresponds to a realizer of A. However, not all realizers are correct programs thus not all realizers represent a proof. In fact, it was revealed by realisability models based on *orthogonality* that the presence of incorrect programs is crucial to give a computational status to correctness.

When introduced by Kleene, realisability was only considered for intuistionnistic logics due to their ‘constructive’ nature, and it is only in 2005 that Jean-Louis Krivine introduced *classical realisability* [3] aiming at extending realisability techniques to classical logic, proposing a model based on orthogonality. Krivine’s construction is based on an extension of the untyped lambda calculus, but, in order to capture a given context (stack) to potentially restore it later, the syntax is not only extended with the call/cc operator but also with a countably infinite set of *stack constants*. As a consequence, (as in Kleene’s realisability) only *some* of the programs represent a classical proof, namely those not containing stack constants. This introduction of "incorrect" terms is essential, as it introduces in the syntax semantic information [4] that can be used to *test* correct (and incorrect) terms. This concept of testing is captured by the definition of an orthogonality relation (here between terms and stacks), which is used to define the interpretation of types (as the set of terms passing a given set of tests).

In parallel with the work of Krivine, similar realisability constructions have been introduced by Jean-Yves Girard in order to interpret Linear Logic. While the orthogonality construction was clearly put forth in Ludics, the ideas and first occurrences can be traced back to the first model of geometry of interaction (GoI) [5], which is restricted to multiplicative linear logic, and interprets proofs as permutations. Later GoI construction took several diverse forms, generalising permutations by operators in a C*-algebra (GOI1 [6], GOI2 [7]), first-order prefix rewriting (GOI3) [8], or von Neumann Algebras (GOI5) [9].

In a series of recent papers [10, 11, 12], Thomas Seiller proposed a combinatorial approach to the Geometry of Interaction, *interaction graphs*, which specialises to all the previous ‘geometries’ of interaction proposed by Girard. It is crucial to note that this work on GOI constructs the types of Linear Logic via a realisabilty method, involving orthogonality within the computational model of interaction graphs. However, proofs are interpreted in these models as abstract objects (generalisations of dynamical systems) which remain far from the general intuition of what a proof is.

This is where our work starts: we extend the use of realizability techniques to Linear Logic in an *untyped variant* of the well known and ‘canonical’ context of proof nets; first to the multiplicative fragment of Linear Logic, and secondly to second order multiplicative Linear Logic. We obtain the results of soundness (e.g. adequacy) and completeness both for MLL and $\mathsf{MLL}^{\maltese}$ – with furthermore assumptions on the interpretation basis. Soundness is also true at the second–order for MLL_2 proofs. Moreover we show that the types constructed by induction for both the multiplicative and second–order preserve the *finite testability* [2]. In particular this is true for the types capturing the proofs of the multiplicative fragment: this is done by encoding the Danos Regnier criterion [13] in $\mathsf{MLL}^{\maltese}$ proofs, we provide, to our knowledge, the first proper

[2]A type $\mathbf{A}$ is finitely testable if there exists a finite set B such that $\mathbf{A} = B^{\perp}$.

proof of the folklore result which states that 'tests of A are proofs of its negation'. We are still investigating how to capture the proofs of the second order multiplicative fragment while remaining finitely testable. We believe this will lead to a novel correctness criterion for second order multiplicative proof structures.

Summary of our work

As a computational model we chose the model of *nets*, a modern formulation – as hypergraphs – of the model of proof structures introduced by Jean Yves Girard in his seminal paper [14]. Informally speaking the nets are hypergraphs constructed by composing the hyperedges, called links, of the figure 1 such that a vertex is the target (resp. source) of at most one link. The conclusion of a net is a vertex that is the source of no link and our hypergraphs are equipped with an order on their conclusion. Nets that have conclusions can *interact* by placing cut links in between their conclusions. Given two nets S and T their interaction is denoted $S :: T$.

Furhtermore, the nets come with a notion of computation which corresponds to cut elimination illustrated in figure 2. Contrary to the original multiplicative proof structures introduced in 1987 by Jean Yves Girard, this rewriting is non–deterministic and not confluent.

The computation gives rise to the notion of orthogonality; two nets S and T are orthogonal whenever their interaction $S :: T$ has *at least one* way to reduce to the net ✠ the daimon link with no output – we then denote $S \perp T$.

Definition 1 (Types)**.** The orthogonal $A^{\perp}$ of a set of multiplicative nets A is defined by $\{P \mid \forall a \in A, P \perp a\}$. A *type* $\mathbf{A}$ is a set of multiplicative nets such that $\mathbf{A}^{\perp\perp} = \mathbf{A}$, or equivalently such that $\mathbf{A} = B^{\perp}$ for some set B.

Given a net S with its conclusion ordered as $p_1 < \cdots < p_n$ for an integer $1 \leq i \leq n$ we denote $S(i)$ the conclusion p_i of S. Furthermore, we let $\mathsf{Pos}(S)$ denote its set of vertices. Given two nets S and T their sum $S + T$ corresponds to the union of their graphs in which the set of links is assumed to be disjoint.

Definition 2 (Construction on types)**.** Given $\mathbf{A}$ and $\mathbf{B}$ two types we define several constructions:

- Their *parallel sum* $\mathbf{A} \parallel \mathbf{B} = \{a + b \mid a \in \mathbf{A}, b \in \mathbf{B}, \mathsf{Pos}(a) \cap \mathsf{Pos}(b) = \emptyset\}^{\perp\perp}$.
- Their *functional composition* $\mathbf{A} \cdot \mathbf{B} = \{S \mid \text{for any } a \in \mathbf{A}^{\perp}, S :: a \in \mathbf{B}\}^{\perp\perp}$.
- The tensor product of two types, $\mathbf{A} \otimes \mathbf{B} = \{a + b + \langle a(1), b(1) \rhd_{\otimes} p\rangle \mid \mathsf{Pos}(a) \cap \mathsf{Pos}(b) = \emptyset, a \in \mathbf{A}, b \in \mathbf{B}\}^{\perp\perp}$.
- The $\parr$–product of two types, $\mathbf{A} \parr \mathbf{B} = (\mathbf{A}^{\perp} \otimes \mathbf{B}^{\perp})^{\perp}$.

Definition 3 (Interpretation Basis)**.** An *interpretation basis* $\mathcal{B}$ is a function that associates to each atomic proposition X a type $[\![X]\!]_{\mathcal{B}}$, the *interpretation* of X, such that

- Each net in $[\![X]\!]_{\mathcal{B}}$ has one conclusion.
- For any atomic proposition X we have $[\![X^{\perp}]\!]_{\mathcal{B}} = [\![X]\!]_{\mathcal{B}}^{\perp}$.

Definition 4 (Realizer of a formula)**.** Given an interpretation basis $\mathcal{B}$, the *interpretation* of a formula is lifted from atomic formulas to any formula and sequents of MLL by induction;

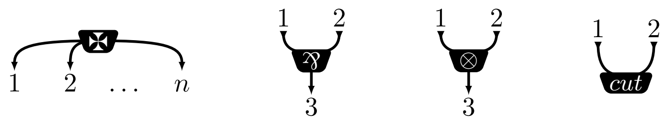
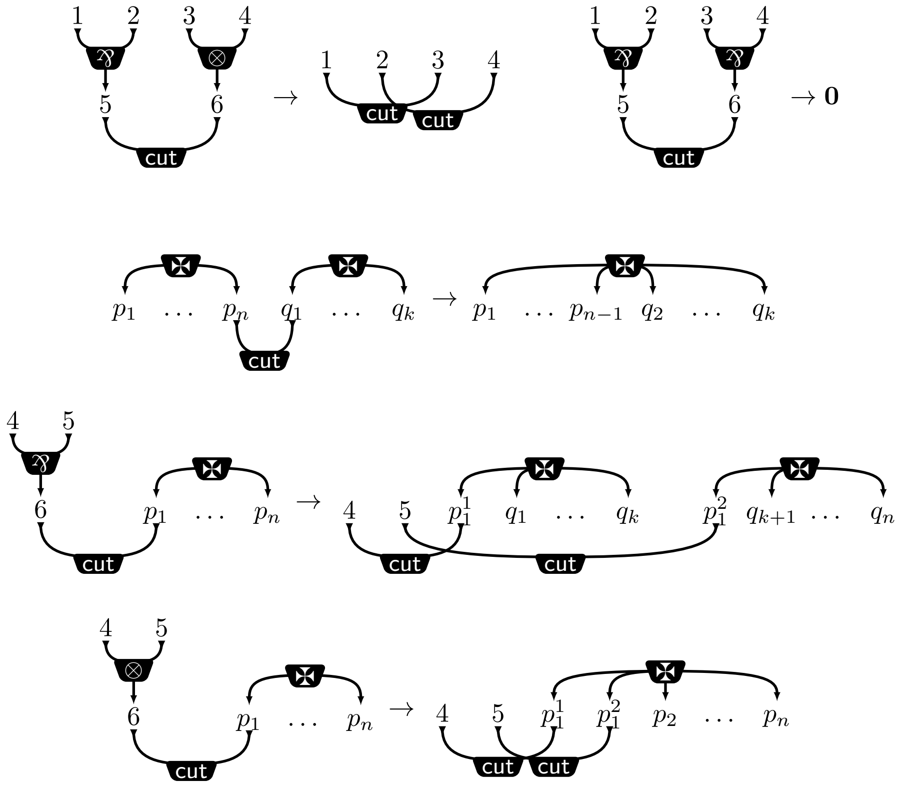


Figure 1: Links defining the class of multiplicative nets. From left to right, they are respectively called daimon link, parr ($⅋$) link, tensor link and cut link.



cut
→
→ 0
p_1 ... p_n q_1 ... q_k
p_1 ... p_{n-1} q_2 ... q_k
p_1^1 q_1 ... q_k p_1^2 q_{k+1} ... q_n
p_1^1 p_1^2 p_2 ... p_n

Figure 2: Rules for the homogeneous cut elimination (in the first two rows) and non homogeneous cut–elimination (in the last two rows). The cut elimination reduction depends on the kind of cut eliminated e.g. the label of the links that are above the inputs of the cut link. The non–homogeneous cut elimination of a daimon against a $⅋$–link is non deterministic, $\{q_1, \dots, q_k\}, \{q_{k+1}, \dots, q_n\}$ is a partition of $\{p_2, \dots, p_n\}$.

$$[\![A \otimes B]\!]_{\mathcal{B}} \triangleq [\![A]\!]_{\mathcal{B}} \otimes [\![B]\!]_{\mathcal{B}} \quad ; \quad [\![A ⅋ B]\!]_{\mathcal{B}} \triangleq [\![A]\!]_{\mathcal{B}} ⅋ [\![B]\!]_{\mathcal{B}} \quad ; \quad [\![A_1, \dots, A_n]\!]_{\mathcal{B}} \triangleq [\![A_1]\!]_{\mathcal{B}} \circ \dots \circ [\![A_n]\!]_{\mathcal{B}}.$$

If there is no ambiguity we relax the notation $[\![\Gamma]\!]_{\mathcal{B}}$ to $[\![\Gamma]\!]$. We denote $S \Vdash_{\mathcal{B}} A_1, \dots, A_n$ whenever $S \in [\![\Gamma]\!]_{\mathcal{B}}$.

Interpretation basis may come with several properties to ensure adequacy or completeness for the logical system considered, namely MLL, MLL$^{✠}$ or MLL$_2$. A basis is:

- *self dual* whenever it maps atomic variables to self dual types $\mathbf{A} \subseteq \mathbf{A}^{\perp}$.
- *approximable* whenever it maps atomic variables to types containing the net made of one daimon link with one conclusion.

Theorem 5 (Adequacy). *Let S be a multiplicative net and Γ be a sequent,*
- *For any basis $\mathcal{B}$; $S \vdash_{\mathsf{MLL}} \Gamma \Rightarrow S \Vdash_{\mathcal{B}} \Gamma$.*
- *For any approximable basis $\mathcal{B}$; $S \vdash_{\mathsf{MLL}^{\maltese}} \Gamma \Rightarrow S \Vdash_{\mathcal{B}} \Gamma$.*

Theorem 6 ($\mathsf{MLL}^{\maltese}$ completeness). *Given some sequent Γ and S a cut–free net and $\mathcal{B}$ a self dual and approximable interpretation basis, if S belongs to $[\![\Gamma]\!]_{\mathcal{B}}$ then S represents a proof of Γ from $\mathsf{MLL}^{\maltese}$.*

Definition 7 (Intersection and union type). Let $\mathcal{B}$ be an interpretation basis, and Ω be a set of types with one output. Given a Γ a sequent of MLL formulas and X a propositional variable the *intersection type* and *union type* on Ω of Γ in X w.r.t. to $\mathcal{B}$ are defined as follow;

$$[\![\bigcap_{X\in\Omega} \Gamma]\!]_{\mathcal{B}} \triangleq \bigcap_{R\in\Omega} [\![\Gamma]\!]_{\mathcal{B}\{X\mapsto R\}} \qquad [\![\bigcup_{X\in\Omega} \Gamma]\!]_{\mathcal{B}} \triangleq \left(\bigcup_{R\in\Omega} [\![\Gamma]\!]_{\mathcal{B}\{X\mapsto R\}}\right)^{\perp\perp}.$$

Theorem 8 (MLL completeness). *Given S a proof like and cut–free net and $\mathcal{B}$ some approximable interpretation basis. If S symmetrically realizes $\bigcap_{X\in\mathcal{V}}[\![\bigcap_{X\in\Omega}\Gamma]\!]_{\mathcal{B}}$ then S is the image of a proof in MLL.*

Definition 9 (realizers of MLL_2). Let $\mathcal{B}$ be an approximable interpretation basis and Ω denote the set of types with one output. Given a formula A of MLL_2 its set of *realizers* is given by the following induction:

$$\begin{array}{ll} [\![A \otimes B]\!] \triangleq [\![A]\!] \otimes [\![B]\!] & [\![\forall X\ A]\!] \triangleq \{S + \langle S(1) \rhd_{\forall} q\rangle \mid S \in [\![\bigcap_{X\in\Omega} A]\!]\} \\ [\![A \parr B]\!] \triangleq [\![A]\!] \parr [\![B]\!] & [\![\exists X\ A]\!] \triangleq \{S + \langle S(1) \rhd_{\exists} q\rangle \mid S \in [\![\bigcup_{X\in\Omega} A]\!]\}^{\perp\perp} \end{array}$$

Theorem 10 (Soundness for MLL_2). *Let $\mathcal{B}$ be an approximable interpretation basis. Given S a proof–like multiplicative second order net. If S represents a proof of the sequent Γ then S belongs to $[\![\Gamma]\!]_{\mathcal{B}}$.*

References

[1] A. Miquel, De la formalisation des preuves à l'extraction de programmes, Habilitation at Université Paris Diderot (2009). URL: https://www.fing.edu.uy/~amiquel/publis/hdr.pdf.

[2] S. C. Kleene, On the interpretation of intuitionistic number theory, The Journal of Symbolic Logic 10 (1945) 109–124. doi:10.2307/2269016.

[3] J.-L. Krivine, Realizability in classical logic, Panoramas et synthèses 27 (2009) 197–229. URL: https://hal.science/hal-00154500.

[4] A. Naibo, M. Petrolo, T. Seiller, On the Computational Meaning of Axioms, Springer International Publishing, Cham, 2016, pp. 141–184. URL: https://doi.org/10.1007/978-3-319-26506-3_5. doi:10.1007/978-3-319-26506-3_5.

[5] J. Girard, Multiplicatives, in: G. Lolli (Ed.), Logic and Computer Science: New Trends and Applications, Rosenberg & Sellier, 1987, pp. 11–34.

[6] J.-Y. Girard, Geometry of interaction 1: Interpretation of system f, in: R. Ferro, C. Bonotto, S. Valentini, A. Zanardo (Eds.), Logic Colloquium '88, volume 127 of *Studies in Logic and the Foundations of Mathematics*, Elsevier, 1989, pp. 221–260. URL: https://www.sciencedirect.com/science/article/pii/S0049237X08702714. doi:https://doi.org/10.1016/S0049-237X(08)70271-4.

[7] J. Girard, Geometry of interaction 2: deadlock-free algorithms, in: P. Martin-Löf, G. Mints (Eds.), COLOG-88, International Conference on Computer Logic, Tallinn, USSR, December 1988, Proceedings, volume 417 of *Lecture Notes in Computer Science*, Springer, 1988, pp. 76–93. URL: https://doi.org/10.1007/3-540-52335-9_49. doi:10.1007/3-540-52335-9_49.

[8] J.-Y. Girard, Geometry of interaction III: accommodating the additives, London Mathematical Society Lecture Note Series, Cambridge University Press, 1995, p. 329–389. doi:10.1017/CBO9780511629150.017.

[9] J. Girard, Geometry of interaction V: logic in the hyperfinite factor, Theor. Comput. Sci. 412 (2011) 1860–1883. URL: https://doi.org/10.1016/j.tcs.2010.12.016. doi:10.1016/j.tcs.2010.12.016.

[10] T. Seiller, Interaction graphs: Graphings, Annals of Pure and Applied Logic 168 (2017) 278–320. doi:10.1016/j.apal.2016.10.007.

[11] T. Seiller, Interaction graphs: Exponentials, Log. Methods Comput. Sci. 15 (2013).

[12] T. Seiller, Interaction graphs: Full linear logic, CoRR abs/1504.04152 (2015). URL: http://arxiv.org/abs/1504.04152. arXiv:1504.04152.

[13] V. Danos, L. Regnier, The structure of multiplicatives, Archive for Mathematical Logic 28 (1989) 181–203. URL: https://doi.org/10.1007/BF01622878. doi:10.1007/BF01622878.

[14] J.-Y. Girard, Linear logic, Theoretical Computer Science 50 (1987) 1 – 101. URL: http://www.sciencedirect.com/science/article/pii/0304397587900454. doi:https://doi.org/10.1016/0304-3975(87)90045-4.