From the proceedings of Trends in Linear Logic and Applications 2021 (TLLA 2021, eds E. Pimentel, Ch. Retoré)
https://hal.archives-ouvertes.fr/TLLA2021
Distributed under a Creative Commons Attribution - Non commercial 4.0 International License

A gentle introduction to Girard's Transcendental Syntax

Boris Eng* Thomas Seiller†‡

Abstract

We present a gentle introduction to the technical content of Girard's Transcendental Syntax suggesting a new framework for the proof theory of linear logic and an alternative understanding of proof-nets. In this framework, we investigate the emergence of logic from a model of computation related to tiling models and logic programs.

1 Introduction

Girard's Transcendental Syntax is the successor of his Geometry of Interaction programme with the ambition of establishing a new framework for proof theory. In this context, logic is grounded on a model of computation related to logic programming and tiling models. Proofs do not come as primitive and elementary objects anymore but as particular computational objects together with a certificate asserting their logical correctness and the soundness of their computational behaviour through cut-elimination. The other usual definitions of logic such as formulas and truth are also reconstructed computationally. The Transcendental Syntax provides a toolbox for a study of proof theory in a more general setting where computation is central.

2 Contributions

The original papers [Gir17, Gir16a, Gir16b, Gir20] presenting the programme give conceptual directions and sketch the outlines for a reconstruction of logic starting from linear logic's proof-nets. However, no true technical development exists yet. Hence, we propose an attempt at formalising the Transcendental Syntax and make its novelties and relationship with other works explicit. In particular,

- we formally define the stellar resolution; the model of computation on which the Transcendental Syntax is based on and prove that it can simulate cut-elimination and logical correctness for MLL(+MIX). We are also working

*LIPN – Université Sorbonne Paris Nord
†CNRS

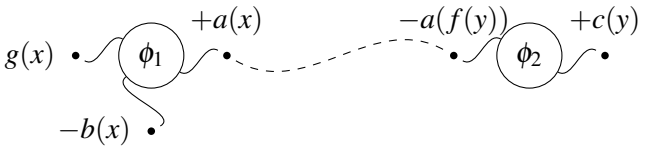
Distributed under a Creative Commons Attribution - Non commercial 4.0 International License

on extensions to Intuitionistic MELL and beyond. Moreover, we make explicit two notions of type/formula in logic. One is inspired from the logical correctness of proof-structures where formulas are simply labels and the other one is inspired from realisability theory [KCHM09, Kle45] where formulas are descriptions of computational behaviours. Both are expressed in the stellar resolution.

- We also relate the stellar resolution to tiling models such as flexible tiles or the abstract tile assembly models, two generalisations of Wang tiles used in DNA computing. This suggests new ideas of typing and implicit computational complexity analysis for tiling-based computation.

3 Technical development

Stellar Resolution The stellar resolution is the model of computation the Transcendental Syntax uses as ground for logic. It is basically a graph-theoretic reformulation of Robinson's resolution with first-order disjunctive clauses [R$^+$65]. It can be related to other resolution-based models [Kow75, Sic76], disjunctive logic programs [Min94] or to answer set programming [Gel08] but it seems that is has never been used as a logic-free and query-free model of computation.



We define *rays* as either first-order terms with a polarised head function symbol called its *colour* (e.g. $+a(x)$ has a as colour) or an unpolarised first-order term (e.g. x or $g(x)$). A *star* is a finite multiset of rays $\phi = [r_1, ..., r_n]$ which can be illustrated as a kind of tile with flexible arms. A *constellation* is a (potentially infinite) multiset of stars written as a sum $\Phi = \phi_1 + ... + \phi_n$ when it is finite.

It is possible to non-deterministically construct sorts of tilings called *diagrams* by connecting occurrences of stars along matchable rays of opposite polarity (e.g. $+a(x)$ and $-a(f(y))$ can have equal underlying terms with $x \mapsto f(y)$). Diagrams can be evaluated by a step-by-step contraction of links with Robinson's resolution rule [R$^+$65]. The *execution* $\mathtt{Ex}(\Phi)$ of a constellation computes all the possible "correct" diagrams and evaluates them in order to form a new constellation. By "correct", we usually mean that we exclude diagrams for which the term unification fails and diagrams with free polarised rays because they corresponds to incomplete/unfinished computations.

This model of computation embodies the old intuition of a theory of interaction behind logic and programming [Abr08] by defining logic from computational interaction between entities of the same kind.

Distributed under a Creative Commons Attribution - Non commercial 4.0 International License

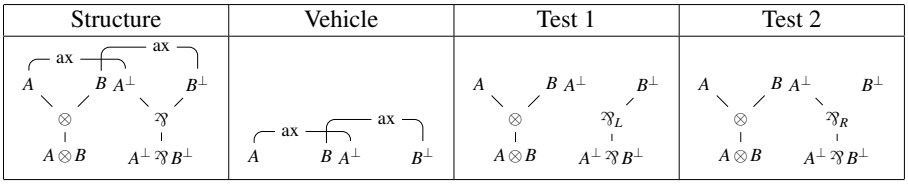


Figure 1: A proof-structure decomposed into a vehicle and its tests.

Geometry of Interaction The Geometry of Interaction sees proofs as general mathematical objects where computation (cut-elimination) is primitive. Nowadays, it refers to a lot of notions such as categorial frameworks for linear logic [HS06, AHS02] or token machines [DR99] but in our case, we refer to Girard's original Geometry of Interaction expressed with operator algebras [Gir89].

The stellar resolution can actually be understood as a generalisation of Girard's flows [Gir95, Bag14], Seiller's interaction graphs [Sei16] but also partitions used as a model for MLL [AM20].

Logical correctness as interactive testing Proof-structures are general hypergraphs linking formulas. Their dynamics (cut-elimination) corresponds to an edge contraction following logical rules represented by their hyperedges.

A proof-structure is made of two independent components. The $⅋/\otimes$ cut-elimination is basically a rewiring pushed to the conclusions of axioms, hence only the axioms are related to the computational side of proofs. The upper part, made of axioms, is called *vehicle*. The logical content is contained in the bottom part (basically a syntax tree of conclusion formulas). It is called the *format* and is constituted of several *tests* which are hypergraphs corresponding to Danos-Regnier switching graphs [DR89]. The Danos-Regnier correctness criterion states that the connexion between the vehicle and any test of the format should be connected and acyclic in order to be called logically correct (Figure 1). In the Transcendental Syntax, we would like to have a broader notion of logical correctness as a certification for constellations which represent a very general idea of computation.

A stellar reconstruction The hyperedges of proof-nets induce a polarity opposing input and output which can easily be represented with stars. For instance, an axiom between atoms $\vdash X, X^\perp$ is represented as the binary star $[+c.p_X(x), +c.p_{X^\perp}(x)]$ but if a ray is not related to a cut then we keep it uncoloured (e.g. $+c.p_X(x)$ becomes $p_X(x)$). A cut on X and $X^\perp$ is translated as $[-c.p_X(x), -c.p_{X^\perp}(x)]$.

The cut rule merges a constellation representing a vehicle Φ_V and a constellation Φ_C representing cuts. Their execution $\mathtt{Ex}(\Phi_V \uplus \Phi_C)$ naturally computes the maximal paths alternating between axioms and cuts which corresponds to the cut-elimination procedure, as expected from the Geometry of Interaction.

The translation is designed in order to reproduce the shape of proof-structures, hence tests $t_1, ..., t_k$ will be translated into constellations $\Phi_{t_1} \uplus ... \Phi_{t_k}$ such that Φ_{t_i}

Distributed under a Creative Commons Attribution - Non commercial 4.0 International License

forms a tiling preserving the shape of t_i. The correctness criterion for a proof-structure $\mathscr{S}$ corresponds to the execution $\mathtt{Ex}(\Phi_V \uplus \Phi_{t_i}) = [p_{A_1}(x), ..., p_{A_n}(x)]$ between a vehicle and a test t_i where $A_1, ..., A_n$ are the conclusions of $\mathscr{S}$. This is only possible when $\Phi_V \uplus \Phi_t$ represents a connected and acyclic switching graph. Since constellations are way more general than proof-structures, we can imagine a more general notion of logical correctness.

Two notions of type in logic It is possible to make explicit two notions of type or formula. In order to do so, we have to choose a relation of orthogonality $\perp$ between constellations representing what we consider a sound interaction. We usually choose the relation in order to capture logical correctness.

- The first kind of typing corresponds to a notion of primitive types. We define (finitely many) tests $\Phi_A := \Phi_1, ..., \Phi_n$ corresponding to a certification for a label A. A constellation Φ is of type A when for each test Φ_i, $\Phi \perp \Phi'$. This generalises the correctness criterion for proof-structures.
- The second kind of typing corresponds to the model of linear logic used in ludics [Gir01] and geometry of interaction which is close to realisability models. In this context, types (called *conducts*) are descriptions of computational behaviours. We define a pre-conduct as a set of constellations $\mathbf{A}$ and $\mathbf{A}^\perp$ as the set of constellations orthogonal to $\mathbf{A}$ (the good partners for A). A conduct is defined as a pre-conduct closed by biorthogonal, i.e. $\mathbf{A} = \mathbf{A}^{\perp\perp}$ (they are the good partners or their good partners). In particular, we can usually consider infinitely many types and subtypes.
- We then require that these two notions are related by an *adequacy lemma* (as in realisability models) stating that the testing for A is sufficient to ensure membership in $\mathbf{A}$ (the conduct corresponding to A). In other words, the tests ensure the expected behaviour. This can also be understood as the inclusion $(\Phi_A)^\perp \subseteq \mathbf{A}$.

4 Results

We showed standard properties for stellar resolution, ensuring a sound computational behaviour. The Turing-completeness follows from an encoding inspired by logic programming with Horn clauses.

Theorem 1 (Confluence)**.** *Let Φ be a constellation. The steps of its execution $\mathtt{Ex}(\Phi)$ can be done in any order with no effect on the result. All results will be equivalent up to renaming of variables.*

Proposition 2 (Turing-completeness)**.** *The stellar resolution can simulate non-deterministic Turing machines.*

We showed that it can simulate both cut-elimination and logical correctness.

Distributed under a Creative Commons Attribution - Non commercial 4.0 International License

Lemma 3 (Simulation of dynamics)**.** *Let $\mathscr{R}$ be a proof-structure such that $\mathscr{R}$ reduces to $\mathscr{S}$ by cut-elimination. We have:*

$$\mathtt{Ex}(\Phi^{\mathrm{ax}}_{\mathscr{R}} \uplus \Phi^{\mathrm{cut}}_{\mathscr{R}}) = \mathtt{Ex}(\Phi^{\mathrm{ax}}_{\mathscr{S}} \uplus \Phi^{\mathrm{cut}}_{\mathscr{S}})$$

where $\Phi^{\mathrm{ax}}_{\mathscr{R}}$ and $\Phi^{\mathrm{cut}}_{\mathscr{R}}$ are respectively the encoding of the vehicle and cuts of $\mathscr{R}$ in the stellar resolution.

Theorem 4 (Stellar correctness criterion)**.** *A proof-structure $\mathscr{S}$ is an MLL proof-net if and only if for all switchings φ. We have:*

$$\mathtt{Ex}(\Phi^{\mathrm{ax}}_{\mathscr{S}} \uplus \Phi^{\mathrm{cut}}_{\mathscr{S}} \uplus \Phi^{\mathrm{test}}_{\mathscr{S},\varphi}) = [p_{A_1}(x), ..., p_{A_n}(x)]$$

where $A_1, ..., A_n$ are the conclusions of $\mathscr{S}$ and $\Phi^{\mathrm{test}}_{\mathscr{S},\varphi}$ is the encoding of the test for the switching φ in the stellar resolution.

We finally showed soundness and completeness in MLL(+MIX) for a class of constellations representing proofs called *proof-like* (as in classical realisability models).

Theorem 5 (Full soundness)**.** *Let $\mathscr{S}$ be an MLL(+MIX) proof-net of conclusion $\vdash \Gamma$. The constellation $\mathtt{Ex}(\Phi^{\mathrm{ax}}_{\mathscr{S}} \uplus \Phi^{\mathrm{cut}}_{\mathscr{S}})$ is proof-like and member of the conduct corresponding to $\vdash \Gamma$.*

Theorem 6 (Full completeness)**.** *If a constellation Φ is proof-like and member of the conduct corresponding to $\vdash \Gamma$, there exists an MLL(+MIX) proof-net $\mathscr{R}$ of conclusion $\vdash \Gamma$ such that its vehicle is Φ.*

5 Current, future and related works

Extension to box-free intuitionistic MELL The cut-elimination has already been studied in the Geometry of Interaction and only requires binary stars [Gir95, Bag14]. The correctness criterion (already sketched by Girard [Gir17]) relies on the mechanisms of stellar resolution which are not present with proof-structures. However, it still has the usual and known drawbacks of proof-nets.

Extension to predicate logic A sketch for a reconstruction of predicate logic is suggested in the third paper of Transcendental Syntax [Gir16b]. Terms are encoded as multiplicative formulas and equality as linear equivalence. In the Transcendental Syntax, predicate logic is considered as part of second order logic because of its implicit quantifications. This may open a new understanding of descriptive complexity where predicate logic is central.

Extension to second order The Transcendental Syntax treats the existential witness in $\exists X.A$ as a constellation part of the vehicle [Gir16b]. In some sense, we consider it as "part of the answer", hard coded in the proof. A proof becomes a pair $(\Phi_V \uplus \Phi_M, \Phi_F)$ where Φ_M is a sort of test corresponding to the formula carried by the existential witness.

Distributed under a Creative Commons Attribution - Non commercial 4.0 International License

Relation with tiling models The stellar resolution generalises the flexible tiling model [JM06, JM05] but also the abstract tile assembly model [Win98] by representing both a tile set and its environment as constellations. This opens the possibility of typing or analysis of computational behaviour/complexity for tiling-based models. Interestingly, a reverse encoding seems to hold: few works suggest the possibility of encoding the dynamics of logic programming, resolution and Horn clauses from DNA computing [Kob99, RKS09, RPdMS14].

Implicit Complexity Theory The model of flows, which can be seen as a restriction of stellar resolution to binary stars, has already been used to characterise complexity classes [BP01, AB14, ABS16]. Since the stellar resolution is more general, we can expect a broader complexity analysis. From the generalisation of flexible tiles, our model might be related to non-deterministic complexity [JM09] but also be related to new complexity methods inspired from Seiller's works [Sei18].

References

[AB14] Clément Aubert and Marc Bagnol. Unification and logarithmic space. In *Rewriting and Typed Lambda Calculi*, pages 77–92. Springer, 2014.

[Abr08] Samson Abramsky. Information, processes and games. *J. Benthem van & P. Adriaans (Eds.), Philosophy of Information*, pages 483–549, 2008.

[ABS16] Clément Aubert, Marc Bagnol, and Thomas Seiller. Unary resolution: Characterizing ptime. In *International Conference on Foundations of Software Science and Computation Structures*, pages 373–389. Springer, 2016.

[AHS02] Samson Abramsky, Esfandiar Haghverdi, and Philip Scott. Geometry of interaction and linear combinatory algebras. *Mathematical Structures in Computer Science*, 12(5):625–665, 2002.

[AM20] Matteo Acclavio and Roberto Maieli. Generalized connectives for multiplicative linear logic. In *28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[Bag14] Marc Bagnol. *On the resolution semiring*. PhD thesis, Aix-Marseille Université, 2014.

[BP01] Patrick Baillot and Marco Pedicini. Elementary complexity and geometry of interaction. *Fundamenta Informaticae*, 45(1-2):1–31, 2001.

Distributed under a Creative Commons Attribution - Non commercial 4.0 International License

[DR89] Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Archive for Mathematical logic*, 28(3):181–203, 1989.

[DR99] Vincent Danos and Laurent Regnier. Reversible, irreversible and optimal λ-machines. *Theoretical Computer Science*, 227(1-2):79–97, 1999.

[Gel08] Michael Gelfond. Answer sets. *Foundations of Artificial Intelligence*, 3:285–316, 2008.

[Gir89] Jean-Yves Girard. Geometry of interaction I: interpretation of system f. In *Studies in Logic and the Foundations of Mathematics*, volume 127, pages 221–260. Elsevier, 1989.

[Gir95] Jean-Yves Girard. Geometry of interaction III: accommodating the additives. *London Mathematical Society Lecture Note Series*, pages 329–389, 1995.

[Gir01] Jean-Yves Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical structures in computer science*, 11(3):301, 2001.

[Gir16a] Jean-Yves Girard. Transcendental syntax II: non-deterministic case. 2016.

[Gir16b] Jean-Yves Girard. Transcendental syntax III: equality. 2016.

[Gir17] Jean-Yves Girard. Transcendental syntax I: deterministic case. *Mathematical Structures in Computer Science*, 27(5):827–849, 2017.

[Gir20] Jean-Yves Girard. Transcendental syntax IV: logic without systems. 2020.

[HS06] Esfandiar Haghverdi and Philip Scott. A categorical model for the geometry of interaction. *Theoretical Computer Science*, 350(2-3):252–274, 2006.

[JM05] Nataša Jonoska and Gregory L McColm. A computational model for self-assembling flexible tiles. In *International Conference on Unconventional Computation*, pages 142–156. Springer, 2005.

[JM06] Nataša Jonoska and Gregory L McColm. Flexible versus rigid tile assembly. In *International Conference on Unconventional Computation*, pages 139–151. Springer, 2006.

[JM09] Nataša Jonoska and Gregory L McColm. Complexity classes for self-assembling flexible tiles. *Theoretical Computer Science*, 410(4-5):332–346, 2009.

Distributed under a Creative Commons Attribution - Non commercial 4.0 International License

[KCHM09] JL Krivine, PL Curien, H Herbelin, and PA Melliès. Interactive models of computation and program behavior. 2009.

[Kle45] Stephen Cole Kleene. On the interpretation of intuitionistic number theory. *The journal of symbolic logic*, 10(4):109–124, 1945.

[Kob99] Satoshi Kobayashi. Horn clause computation with dna molecules. *Journal of Combinatorial Optimization*, 3(2):277–299, 1999.

[Kow75] Robert Kowalski. A proof procedure using connection graphs. *Journal of the ACM (JACM)*, 22(4):572–595, 1975.

[Min94] Jack Minker. Overview of disjunctive logic programming. *Annals of Mathematics and Artificial Intelligence*, 12(1-2):1–24, 1994.

[R$^+$65] John Alan Robinson et al. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.

[RKS09] Tom Ran, Shai Kaplan, and Ehud Shapiro. Molecular implementation of simple logic programs. *Nature Nanotechnology*, 4(10):642–648, 2009.

[RPdMS14] Alfonso Rodríguez-Patón, Iñaki Sainz de Murieta, and Petr Sosík. Dna strand displacement system running logic programs. *Biosystems*, 115:5–12, 2014.

[Sei16] Thomas Seiller. Interaction graphs: Full linear logic. In *2016 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10. IEEE, 2016.

[Sei18] Thomas Seiller. Interaction graphs: Non-deterministic automata. *ACM Transactions on Computational Logic (TOCL)*, 19(3):1–24, 2018.

[Sic76] Sharon Sickel. A search technique for clause interconnectivity graphs. *IEEE Transactions on Computers*, (8):823–835, 1976.

[Win98] Erik Winfree. *Algorithmic self-assembly of DNA*. PhD thesis, Citeseer, 1998.