A gentle introduction to Girard's Transcendental Syntax for the linear logician

Boris Eng

▶ **To cite this version:**

Boris Eng. A gentle introduction to Girard's Transcendental Syntax for the linear logician. 2021. hal-02977750v5

HAL Id: hal-02977750

https://hal.science/hal-02977750v5

Preprint submitted on 4 May 2021 (v5), last revised 3 Apr 2022 (v7)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A gentle introduction to Girard's Transcendental Syntax for the linear logician

Version 5

Boris Eng

Girard's Transcendental Syntax suggests a new framework for proof theory where logic (proofs, formulas, truth, ...) is no more primitive but computation is. Logic is grounded on a model of computation I call "stellar resolution" which is basically logic-free Robinson's first-order clausal resolution with a dynamics related to tiling models. This model naturally encodes the cut-elimination for proof-structures. By using realisability techniques for linear logic, it is possible to reconstruct formulas/types and logical correctness in order to obtain models of linear logic. Girard's philosophical justification of these works comes from Kantian inspirations: the Transcendental Syntax appears as a way to talk about the "conditions of possibility of logic", that is the conditions from which logical constructions emerge out of the meaningless (computation). We illustrate this foundational project with a reconstruction of Intuitionistic MELL(+MIX) and describe few other novelties such as the treatment of second order and Girard's logical constants ㄱ (fu) and ㅎ (wo).

Contents

1 Introduction: an alternative foundation for proof theory

The Transcendental Syntax is a programme initiated by Girard which can be seen as the successor of his previous Geometry of Interaction (GoI) programme [15, 14, 13, 16, 18, 19] studying linear logic from the mathematics of the cut-elimination. They both share the same goal of giving a fully computational foundation for logic, which is no more primitive, is reconstructed from a computational model. One may see it as a kind of reverse engineering of linear logic. Unlike other similar approaches (like type theory or classical realisability), it starts from the assumption that a reconstruction of logic should begin from a reconstruction of linear logic from its proof-nets.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B, B^\perp}\,\mathrm{ax} \qquad \dfrac{}{\vdash A, A^\perp}\,\mathrm{ax}}{\vdash A^\perp, B^\perp, B \otimes A}\,\otimes}{\vdash A^\perp \parr B^\perp, B \otimes A}\,\parr}{\vdash (A^\perp \parr B^\perp) \parr (B \otimes A)}\,\parr$$

One ambition is to explain why does it mean when we write an innocent proof in

the sequent calculus. In particular, we are interested in making implicit assumptions computationally explicit, to "put everything on the table" in some sense.

When we materialise a proof, we present a computational object which can be evaluated by the cut-elimination procedure. We assume that this procedure will terminate and work as we would like to but also that a proof terminates by connecting dual formulas. Actually, we can even forget formulas since they are just labels with no real importance.

In the Transcendental Syntax, proofs do not exist by themselves but as hybrid objects $(\Phi_0, \Phi_1 \uplus ... \uplus \Phi_n)$ where the Φ_i are interactive computational objects and $\Phi_1, ..., \Phi_n$ represent tests asserting the logical correctness of Φ_0 (which are usually implicit and not part of the system). We then require that Φ_0 and $\Phi_1, ..., \Phi_n$ have a sound pairwise interaction (yet to be defined). Following this idea, we would like to establish a logic without rules, axioms, logical primitives or even system.

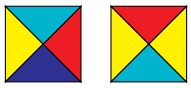
2 Stellar Resolution

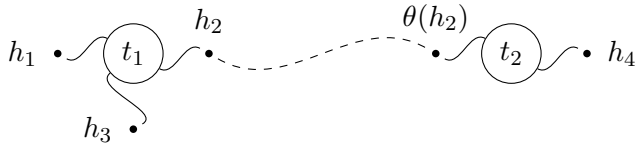
The stellar resolution is the computational ground selected by Girard. In fact, this is not a necessary choice. Other choices can be made but this one is especially convenient and comes from a generalisation of flows [16] coming from the GoI.

We use the name "stellar" for Girard's terminology of *stars and constellations* and "resolution" for its similarities with other resolution-based models [28, 40].

For pedagogical purposes, we describe our model of computation as a generalisation of tiling models and show that it behaves like a kind of logic programming language.

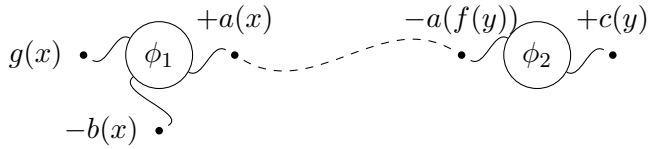
2.1 Tiling models

Wang tiles [42]  We consider bricks with four faces for the four directions in the plane $\mathbf{Z}^2$. We stick the faces together if they have the same colour. We are usually interested in making tilings by reusing the tiles as many times as possible. This is well-known model and a good introduction to tiling-based computation.



Flexible tiles [25, 24] Instead of imposing planarity to tilings, we can allow tiles to have flexible sides selected from set of *sticky-ends types* H. We connect the sticky-ends w.r.t. an involution θ defining complementarity. Surprisingly, this model is able to simulate rigid tiles such as Wang tiles [25].

2.2 Stars and constellations



The stellar resolution can be understood as a flexible tiling model with terms called *rays* as sticky-ends. A ray can either be polarised with a head symbol called *colour* (e.g $+a(x)$ or $-a(f(y))$ where a is a colour where t is any term) or unpolarised (e.g x or $g(x)$).

A *star* (tile) is a finite multiset of rays $\phi = [r_1, ..., r_n]$ and a *constellation* $\Phi = \phi_1 + ... + \phi_m$ (tile set / program) is a (potentially infinite) multiset of stars. We consider stars to be equivalent up to renaming and no two stars within a constellation share variables: these variables are local in the sense of usual programming.

By using occurrences of stars from a given constellation, we connect them along their matchable rays of opposite polarity in order to construct tilings called *diagrams*.

We get the following correspondence of terminology:

Tiles	**Stellar Resolution**
Sticky-end	Ray $r = +a(t), -a(t), t$
Tile	Star $\phi = [r_1, ..., r_n]$
Tile set	Constellation $\Phi = \phi_1 + ... + \phi_m$
Tiling	Diagram

Example 1. *We encode two typical logic programs as constellations. We write s^n for n applications of the symbol s in order to encode natural numbers. A colour corresponds to a predicate and the polarity represents the distinction input/output or hypothesis/conclusion. An unpolarised ray cannot interact with other rays.*

```
add(0, y, y).
add(s(x), y, s(z)) :- add(x, y, z).
?add(s^n(0), s^m(0), r).
```

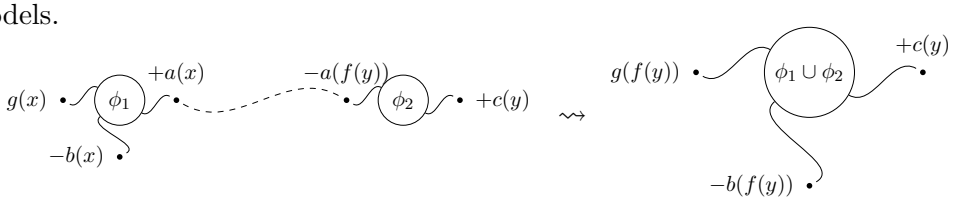
$$\Phi_{\mathbf{N}}^{n,m} = [+add(0, y, y)] + [-add(x, y, z), +add(s(x), y, s(z))] + [-add(s^n(0), s^m(0), r), r]$$

```
parent(d, j). parent(z, d). parent(s, z). parent(n, s).
ancestor(x, y) :- parent(x, y).
ancestor(x, z) :- parent(x, y), ancestor(y, z).
?ancestor(j, r).
```

$$\Phi_{family} = [+parent(d, j)] + [+parent(z, d)] + [+parent(s, z)] + [+parent(n, s)] + [+ancestor(x, y), -parent(x, y)] + [+ancestor(x, z), -parent(x, y), -ancestor(y, z)] + [-ancestor(j, r), r]$$

2.3 Evaluation of constellations

We are now interested in tiling evaluation which is not standard in the theory of tiling models.



We first define an evaluation of diagrams called *actualisation*. If stars are understood as kind of molecules, then evaluating diagrams corresponds to triggering the actual interaction between the stars along their connected rays, thus making a kind of chemical reaction happen and propagate to the non-involved entities. Another way to see it is to solve constraints between the connected rays and propagating the solution to the free (unconnected) rays (which survive to the evaluation).

There are two equivalent ways of contracting diagrams into a star by observing that each edge defines a unification problem (equation between terms):

Fusion We can reduce the links step-by-step by solving the underlying equation, producing a solution θ. The two linked stars will merge by making the connected rays disappear. The substitution θ is finally applied on the rays of the resulting star. This is Robinson's resolution rule.

Actualisation The set of all edges defines a big unification problem. The solution θ of this problem is then applied on the star of free rays.

In order to get a successful evaluation we need our diagrams to satisfy few properties (only the first and last ones are mandatory). They usually need to be:

Connected because otherwise, we would always have infinitely many diagrams by repeating isolated stars (notice that it ensures a reduction into a single star);

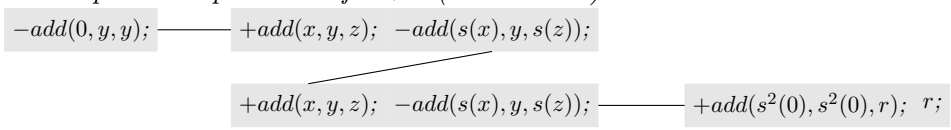
Open meaning that we have unconnected rays because otherwise we would get the empty star which is usually not interesting (no visible output). However, this condition can relaxed in some cases.

Saturated meaning that it is impossible to extend the diagram with more occurrences of stars. It represents a kind of maximal/complete computation. We usually consider the more general exclusion of diagrams with free coloured rays since they represent incomplete computations in some sense (see subsection 3.1).

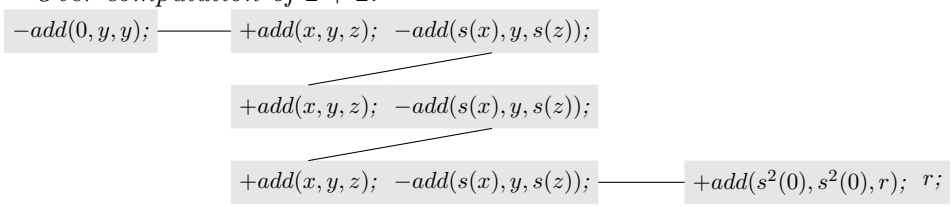
Correct meaning that the actualisation does not fail (no contradiction during conflict resolution).

Example 2 (diagrams for unary addition)**.** *Partial computation of* $2+2$ *(0 recursion):*
$-add(0,y,y);$ ——— $+add(x,y,z);\ -add(s(x),y,s(z));$ ——— $+add(s^2(0),s^2(0),r);\ r;$

Complete computation of $2+2$ (1 recursion):

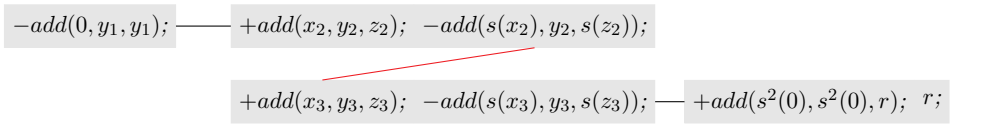


Over computation of $2+2$:

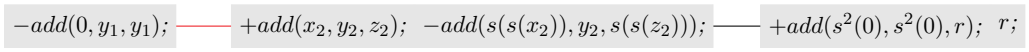


Note that in the case of $\Phi^{n,m}_{\mathbf{N}}$, there is infinitely many saturated diagrams but only one is correct: the one corresponding to a successful computation of $n+m$. Also notice that the maximal size of diagrams naturally corresponds to the time complexity of constellations.

Example 3 (fusion)**.** *The full fusion of the diagram representing a complete computation of $2+2$ from example 2 is described below (we make the exclusion of variable explicit for clarity):*

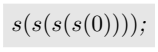


$$\downarrow \theta = \{x_3 \mapsto s(x_2), y_3 \mapsto y_2, z_3 \mapsto s(z_2)\}$$



$$\downarrow \theta = \{x_2 \mapsto 0, z_2 \mapsto y_2\}$$

$$\downarrow \theta = \{y_2 \mapsto s(s(0)), r \mapsto s(s(s(s(0))))\}$$



Example 4 (actualisation)**.** *If we take the diagram δ representing a complete computation in the example 2, it generates the following problem:*

$$\mathcal{P}(\delta) = \{add(0,y_1,y_1) \stackrel{?}{=} add(x_2,y_2,z_2), add(s(x_2),y_2,s(z_2)) \stackrel{?}{=} add(x_3,y_3,z_3),$$
$$add(s(x_3),y_3,s(z_3)) \stackrel{?}{=} add(s^2(0),s^2(0),r)\}$$

which is solved by a unification algorithm such as the Montanari-Martelli algorithm [31] in order to obtain a finale substitution:

$$\rightarrow^* \{x_2 \stackrel{?}{=} 0, y_2 \stackrel{?}{=} y_1, z_2 \stackrel{?}{=} y_1, x_3 \stackrel{?}{=} s(x_2), y_2 \stackrel{?}{=} y_3, z_3 \stackrel{?}{=} s(z_2),$$

$$s(x_3) \stackrel{?}{=} s^2(0), y_2 \stackrel{?}{=} s^2(0), s(z_3) \stackrel{?}{=} r\}$$

$$\rightarrow^* \{y_2 \stackrel{?}{=} y_1, z_2 \stackrel{?}{=} y_1, x_3 \stackrel{?}{=} s(0), y_2 \stackrel{?}{=} y_3, z_3 \stackrel{?}{=} s(z_2), s(x_3) \stackrel{?}{=} s^2(0), y_2 \stackrel{?}{=} s^2(0), s(z_3) \stackrel{?}{=} r\}$$

$$\rightarrow^* \{z_2 \stackrel{?}{=} y_1, x_3 \stackrel{?}{=} s(0), y_1 \stackrel{?}{=} y_3, z_3 \stackrel{?}{=} s(z_2), s(x_3) \stackrel{?}{=} s^2(0), y_2 \stackrel{?}{=} s^2(0), s(z_3) \stackrel{?}{=} r\}$$

$$\rightarrow^* \{x_3 \stackrel{?}{=} s(0), y_1 \stackrel{?}{=} y_3, z_3 \stackrel{?}{=} s(y_1), s(x_3) \stackrel{?}{=} s^2(0), y_1 \stackrel{?}{=} s^2(0), s(z_3) \stackrel{?}{=} r\}$$

$$\rightarrow^* \{y_1 \stackrel{?}{=} y_3, z_3 \stackrel{?}{=} s(y_1), s(s(0)) \stackrel{?}{=} s^2(0), y_1 \stackrel{?}{=} s^2(0), s(z_3) \stackrel{?}{=} r\}$$

$$\rightarrow^* \{z_3 \stackrel{?}{=} s(y_3), s(s(0)) \stackrel{?}{=} s^2(0), y_3 \stackrel{?}{=} s^2(0), s(z_3) \stackrel{?}{=} r\}$$

$$\rightarrow^* \{s(s(0)) \stackrel{?}{=} s^2(0), y_3 \stackrel{?}{=} s^2(0), s(s(y_3)) \stackrel{?}{=} r\}$$

$$\rightarrow^* \{y_3 \stackrel{?}{=} s^2(0), s(s(y_3)) \stackrel{?}{=} r\}$$

$$\rightarrow^* \{s(s(s^2(0))) \stackrel{?}{=} r\}$$

$$\rightarrow^* \{r \stackrel{?}{=} s(s(s^2(0)))\}$$

The solution of this problem is the substitution $\theta = \{r \mapsto s^4(0)\}$ which is applied on the star of free rays $[r]$. The result $[s^4(0)]$ of this procedure is called the actualisation *of δ.*

$$\Phi \quad \stackrel{\text{generates}}{\longrightarrow} \quad \cup_{k=0}^{\infty} D_k \quad \stackrel{\text{actualises}}{\longrightarrow} \phi_1 + ... + \phi_n$$

The *execution* or *normalisation* $\texttt{Ex}(\Phi)$ of a constellation Φ constructs the set of all possible correct saturated diagrams and actualises them all in order to produce a new constellation called the *normal form*.

In logic programming, we can interpret the normal form as a subset of the application of resolution operator [30]. It non-deterministically computes all the "maximal/complete" inferences possible.

If the set of correct saturated diagrams is finite (or the normal form is a finite constellation), the constellation is said to be *strongly normalising*.

Example 5. *For $\Phi_{\mathbf{N}}^{2,2}$ (example 1), one can check that we have $\texttt{Ex}(\Phi_{\mathbf{N}}^{2,2}) = [s^4(0)]$ because only the complete computation of example 2 succeed and all other saturated diagrams represents partial or over computations and fail.*

3 Encoding of some models of computation

We provide concrete examples of how our model compute. It seems that it is especially convenient for models of computation which are also dynamical systems and compute by local interactions between independent objects.

Two interesting points which are central to computing with stellar resolution are made clear by the below examples:

- the stellar resolution corresponds to a general notion of automata due to the fact that the space of terms corresponds to a state space and that diagrams corresponds to hypergraphs on that space. It also make the correspondence between tiling models and automata direct (some links were already investigated [41]).
- models of computation dependent of external definitions (an evaluation function or an environment) are translated into two dual constellations, one corresponding to the model and the other corresponding to its environment. We require that they interact as expected.

3.1 Non-deterministic finite automata

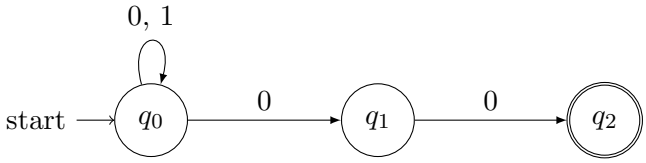
The idea is to represent the transitions by binary bipolar stars.

Let Σ be an alphabet and $w \in \Sigma^*$ a word. If $w = \varepsilon$ then $w^\bigstar = [+i(\varepsilon)]$ and if $w = c_1...c_n$ then $w^\bigstar = [+i(c_1 \cdot (... \cdot (c_n \cdot \varepsilon)))]$. We use the binary function symbol $\cdot$ which is right-associative.

Let $A = (\Sigma, Q, Q_0, \delta, F)$ be a non-deterministic finite automata. We define its translation $A^\bigstar$:

- for each $q_0 \in Q_0$, we have $[-i(w), +a(w, q_0)]$.
- for each $q_f \in F$, we have $[-a(\varepsilon, q_f), accept]$.
- for each $q \in Q, c \in \Sigma$ and for each $q' \in \delta(q, c)$ with $c \in \Sigma$, we have the star $[-a(c \cdot w, q), +a(w, q')]$.

For instance, the following automaton A accepting binary words ending by 00:



is translated as:

$$A^\bigstar = [-i(w), +a(w, q_0)] + [-a(\epsilon, q_2), accept] +$$

$$[-a(0 \cdot w, q_0), +a(w, q_0)] + [-a(1 \cdot w, q_0), +a(w, q_0)] +$$

$$[-a(0 \cdot w, q_0), +a(w, q_1)] + [-a(0 \cdot w, q_1), +a(w, q_2)]$$

The set of saturated correct diagrams corresponds to the set of non-deterministic runs. With the word $[+i(0 \cdot 0 \cdot 0)]$ only the run $q_0q_0q_1q_2$ leads to the finale state. The corresponding diagram will actualise into $[accept]$. The other correct diagrams corresponds to incomplete computations and are excluded. We get $\mathtt{Ex}(A^\bigstar + [+i(0 \cdot 0 \cdot 0)]) = [accept]$, meaning that the word is accepted.

This idea can easily be extended to pushdown automata. We briefly illustrate the idea: the star $[-a(1 \cdot w, 0 \cdot s), +a(w, s)]$ corresponds to checking if we read 1 and that 0 is on the top of the stack and if so, we remove it. Since we manipulate terms and that Turing machines can be seen as pushdown automata extended with two stacks, the extension to tree automata and Turing machines is direct.

Also remark that there is no explicit flow of computation and that paths are constructed non-deterministically in an asynchronous way.

3.2 Boolean circuits

The idea is to first encode an hypergraph representing the structure of a boolean circuit then to connect the resulting constellation to another one containing the implementation of logic gates (a kind of semantic environment).

$$VAR(Y,i) := [-val(x), Y(x), +c_i(x)]$$
$$SHARE(i,j,k) := [-c_i(x), +c_j(x), +c_k(x)]$$
$$AND(i,j,k) := [-c_i(x), -c_j(y), -and(x,y,r), +c_k(r)]$$
$$OR(i,j,k) := [-c_i(x), -c_j(y), -or(x,y,r), +c_k(r)]$$
$$NEG(i,j) := [-c_i(x), -neg(x,r), +c_j(r)]$$
$$CONST(k,i) := [+c_i(k)] \quad QUERY(k,i) := [+c_i(k), R(k)]$$

where i, j, k are encodings of natural numbers representing identifiers. We also have a star $VAR(Y,i)$ for each variable Y we want as input in our boolean circuit.

We consider the following constellation representing a kind of "module" (as in any programming language) providing the definitions of propositional logic:

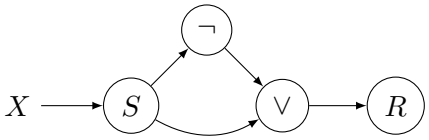
$$\Phi^{\mathcal{PL}} = [+val(0)] + [+val(1)] + [+neg(0,1)] + [+neg(1,0)] +$$
$$[+and(0,0,0)] + [+and(0,1,0)] + [+and(1,0,0)] + [+and(1,1,1)]$$
$$[+or(0,0,0)] + [+or(0,1,1)] + [+or(1,0,1)] + [+or(1,1,1)]$$

We can observe that by changing the module, we can adapt the model and obtain arithmetic circuits (possibly with alternative implementations). Notice that syntax and semantics live in the same world (they are represented as objects of the same kind) and can interact.

We use the star $CONST(k)$ to force a value on the input and the star $QUERY(k)$ to ask for a particular output. For instance $QUERY(1)$ asks for satisfiability.

For instance, we execute the constellation representing $X \vee \neg X$ where $\underline{n}$ is an encoding of natural number:

$$\Phi_{em} = VAR(x, \underline{0}) + SHARE(\underline{0}, \underline{1}, \underline{2}) + NEG(\underline{2}, \underline{3}) + OR(\underline{1}, \underline{3}, \underline{4}) + QUERY(1, \underline{4})$$



The constellation $\Phi_{em} + \Phi^{\mathcal{PL}}$ will generate two diagrams: one corresponding to the input 0 and another one for 1. We obtain $\mathtt{Ex}(\Phi_{em} + \Phi^{\mathcal{PL}}) = [X(0), R(1)] + [X(1), R(1)]$ stating that for the two valuations $x \mapsto 0$ and $x \mapsto 1$, the circuit output $R(1)$.

3.3 Comments on the model

This model of computation embodies the idea of computation-as-interaction [1]. It is a very primitive model of computation which is based on our intuition of space. The rays can be seen as wires we can divide (for instance $f(x)$ into subwires $f(\mathtt{l} \cdot x)$ and $f(\mathtt{r} \cdot x)$) and thus enjoy a topological/geometrical interpretation.

Relationship with logic programming At first, our model seems identical to Robinson's first-order resolution using disjunctive clauses. The difference is that our model is purely computational (no reference to logic) and that we use it for a different purpose (no interest in reaching the empty clause but rather the set of atoms we can infer). Although this dynamics is well-known it seems that it has never been used that way as a logic-free model of computation by its own. Moreover, our model will be extended with internal colours and possibly additional features in the future, thus justifying the use of a new name.

Relationship with tiling models The stellar resolution generalises *flexible tiles* [25, 24], a model of computation coming from DNA computing [43]. This model is itself able to simulate usual tiling-based models of computation such as Wang tiles [42] or abstract tile assembly models [32] by encoding both the tile set and its environment by constellations. From our realisability construction, one can imagine methods of typing and implicit complexity analysis of these models.

Relationship with the GoI and complexity We can generalise flows [16, 7] and interaction graphs [35, 37] which have an applications in implicit computational complexity [8, 5, 6, 39]. Some of these works encode some notions of automata and it seems that the stellar resolution is linked to a very general notion of automata seen as non-deterministic tiling [41].

4 Geometry of Interaction for MLL

The GoI began with a mathematical study of the cut-elimination procedure through the use of infinite-dimensional spaces and operator algebras in order to handle the non-linearity of full linear logic. In the third article of GoI [16], Girard introduced a simplification based on first-order unification: the model of flows [7] which is basically unary first-order resolution, well-known among computer scientists. The stellar resolution is simply an extension of this model which, unlike flows, is able to speak about logical correctness in a satisfactory way. The use of terms and unification is especially convenient because it internalises the infinity present in the GoI ($f(x)$ is matchable with $f(t)$ for any term t).

Technically speaking, the GoI studies generalisations of proof-nets by keeping their essential parts and reconstructing linear logic from the computational content of the cut-elimination procedure.

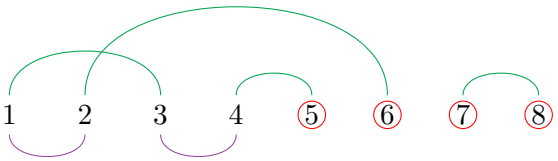
Note that the term "geometry of interaction" also refers to categorical frameworks for linear logic [23, 2] but also combinatorial ones with the token machine [11, 4]. In this document, we are only interested in Girard's original programme [14, 13, 16, 18, 19].

We explain how to encode MLL proof-structures and to simulate both MLL cut-elimination and logical correctness (by the Danos-Regnier criterion) as a first step towards a full reconstruction of linear logic and more especially intuitionistic MELL in the document.

4.1 Cut-elimination and permutations

When considering cut-elimination, axioms induce a permutation on the atoms conclusion of axiom rules. As in ludics [17], we can call these atoms *loci* (plural of *locus*) and they represent kind of physical locations within a proof. We can actually forget the formulas which are simply labels and only see abstract physical locations.

The $\parr/\otimes$ cuts can be seen as administrative/inessential cuts since all they do is basically a rewiring on the premises of the $\parr$ and $\otimes$ nodes connected together. Therefore, such a cut can be seen as a permutation/edge between two loci. We observe that the loci are the "support of the proof"; the locations where logical interaction takes place.



Seiller [35] studied the GoI through juxtaposition of graphs where cut-elimination becomes the computation of maximal alternating paths. In this setting, proof-nets simply speaks about locations and paths between them and truth/correctness is about cycles and connectivity.

The stellar resolution generalises this idea by encoding hypergraphs representing proof-structures. In the case of cut-elimination, we only need binary stars representing graph edges. The above graph becomes:

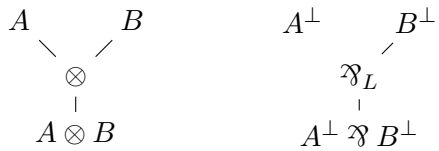
$$\Phi = [+c(1), -c(3)] + [+c(4), 5] + [+c(2), 6] + [7, 8]$$
$$[-c(1), -c(2)] + [-c(3), -c(4)]$$

Notice that we only use the colours $+c$ for the locations which interact with the cuts. We have $\mathtt{Ex}(\Phi) = [5, 6] + [7, 8]$ which corresponds to the expected normal form (set of maximal paths). Notice that the matching is exact, hence the actualisation is a trivial contraction of a unique diagram (since no non-deterministic choice is involved).

4.2 Correctness and partitions

If we want to handle correctness in a satisfactory and natural way for MLL, we have to shift to partitions instead of permutations [12, 3]. Permutations can still be retrieved: a permutation $\{x_1 \mapsto y_1, ..., x_n \mapsto y_n\}$ on $X \subseteq \mathbf{N}$ representing a proof naturally induces a partition of binary sets $\{\{x_1, y_1\}, ..., \{x_n, y_n\}\}$ in X.

A Danos-Regnier switching induces a partition depending on how it separates or groups atoms:



The above switching corresponds to the partition $\{\{1, 2\}, \{3\}, \{4\}\}$. Partitions are related by orthogonality: two partitions are orthogonal if the graph constructed with sets as nodes and where two nodes are adjacent whenever they share a common value is a tree. Testing an axiom-partition "block" against several switching-partitions "blocks" is sufficient to speak about logical correctness.

Since stars are not limited to binarity, we can naturally represent general partitions by constellations: $[-c(1), -c(2)] + [-c(3)] + [-c(4)]$ (notice the negative polarity in order to allow connexion with axioms). However, in this case, all diagrams are closed (no free rays). We have to specify where the conclusions are located: $[-c(1), -c(2), A \otimes B] + [-c(3)] + [-c(4), A^\perp \parr B^\perp]$.

We now consider a constellation representing axioms all coloured with $+c$ in order to allow testing with switchings:

$$\Phi_A = [+c(1), +c(3)] + [+c(2), +c(4)]$$

$$\Phi_{\parr}^L = [-c(1), -c(2), A \otimes B] + [-c(3)] + [-c(4), A^\perp \parr B^\perp]$$

We have $\mathtt{Ex}(\Phi_A \uplus \Phi_{\parr}^L) = [A \otimes B, A^\perp \parr B^\perp]$ which is the star of conclusions. In that case, we say that Φ_A passes the test $\Phi_{\parr}^L$.

The Danos-Regnier criterion is reformulated as follows: "a constellation representing a proof-structure is correct if and only if its execution against all the constellations representing its switchings produces the star of its conclusions".

5 Interpretation of multiplicative linear logic

5.1 The computational content of multiplicatives

We now consider a more general setting by using terms containing variables as rays instead of constants. This is not especially useful for MLL but will be crucial for the exponentials or further extensions (for instance first and second order linear logic).

We set a basis of representation $\mathbb{B}$ with unary symbols p_A for all formulas of MLL, and constants $\mathtt{l}, \mathtt{r}$ to represent the address of a locus relatively to the tree structure of the lower part of a proof-structure. We use a right-associative binary symbol $\cdot$ to glue constants together. Any other isomorphic basis can be considered as well.

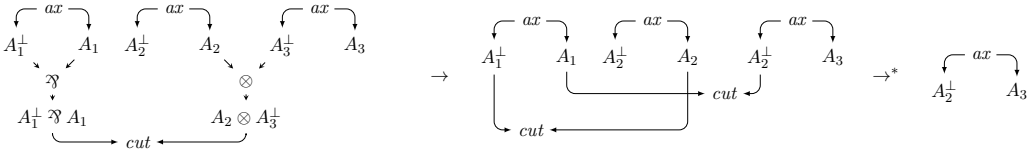
To simulate the dynamics of cut-elimination we translate the axioms and the cuts into stars:

1. A locus A becomes a ray $+c.p_A(t)$ if it is involved with a cut or $p_A(t)$ otherwise, where t represents the "address" of A relatively to the conclusions of the proof-structure (without considering cuts). We use and encoding of the path from a conclusion to A in the tree corresponding to the lower part of the proof-structure. The colour $+c$ stands for "positive computation".

2. An axiom becomes a binary star containing the translations of its loci as described above. For instance, the axiom $A, A^\perp$ becomes $[p_A(x), p_{A^\perp}(x)]$.

3. A cut between A and $A^\perp$ becomes a binary star $[-c.p_A(x), -c.p_B(x)]$ coloured with $-c$ for "negative computation". This colour of opposite polarity allows connexion with the axioms.

Example 6. *We encode the following cut-elimination $\mathcal{S} \to^* \mathcal{S}'$ of MLL proof-structures:*

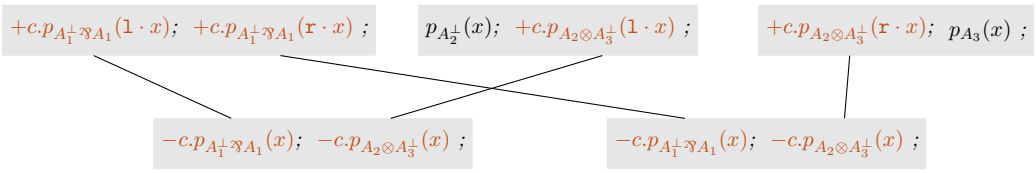


The address of $A_1^\perp$ is $p_{A_1^\perp \parr A_1}(\mathtt{l} \cdot x)$ because it is located on the left-hand side of $A_1^\perp \parr A_1$. The address of $A_3^\perp$ is $p_{A_2 \otimes A_3^\perp}(\mathtt{r} \cdot x)$ and the one for A_3 is $p_{A_3}(x)$. The proof-structure $\mathcal{S}$ is encoded as:

$$[+c.p_{A_1^\perp \parr A_1}(\mathtt{l} \cdot x), +c.p_{A_1^\perp \parr A_1}(\mathtt{r} \cdot x)] + [p_{A_2^\perp}(x), +c.p_{A_2 \otimes A_3^\perp}(\mathtt{l} \cdot x)] +$$

$$[+c.p_{A_2 \otimes A_3^\perp}(\mathtt{r} \cdot x), p_{A_3^\perp}(x)] + [-c.p_{A_2 \otimes A_3^\perp}(x), -c.p_{A_2 \otimes A_3^\perp}(x)]$$

The only correct saturated diagram is:



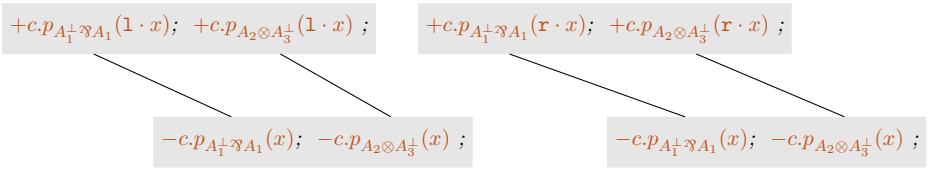
The matching is exact (since no non-deterministic choice involved) so the connected rays are removed and we end up with $[p_{A_2^\perp}(x), p_{A_3}(x)]$ *corresponding to* $\mathcal{S}'$*, as expected.*

Example 7. *If we have the following reduction* $\mathcal{S} \to \mathcal{S}'$ *instead:*

The constellation corresponding to $\mathcal{S}$ *is*

$$[+c.p_{A_1^\perp \parr A_1}(\mathtt{l}\cdot x), +c.p_{A_2\otimes A_3^\perp}(\mathtt{l}\cdot x)] + [+c.p_{A_2\otimes A_2^\perp}(\mathtt{r}\cdot x), +c.p_{A_1^\perp \parr A_1}(\mathtt{r}\cdot x)] +$$
$$[-c.p_{A_1^\perp \parr A_1}(x), -c.p_{A_2\otimes A_3^\perp}(x)]$$

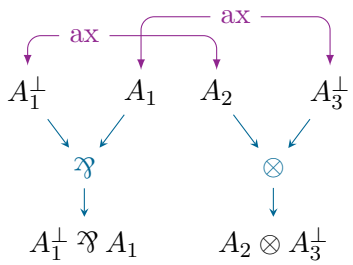
When trying to make a saturated diagram by following the shape of the proof-structure, we end up with:



which is a closed diagram (thus incorrect), hence the normal form is the empty constellation.

5.2 The logical content of multiplicatives

We translate the correctness criterion of Danos-Regnier [10] by an encoding of hypergraphs (representing proof-structures) into constellations.



Any proof-structure can be seen as the sum of two components:

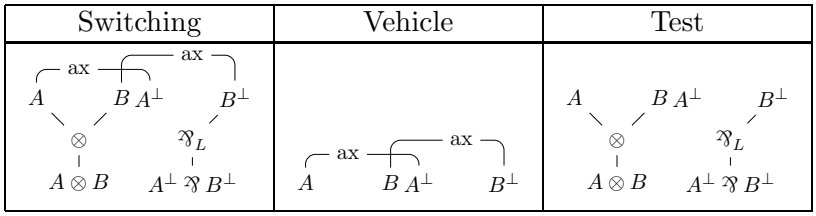


Figure 1: The vehicle of a proof-structure and a test corresponding to a Danos-Regnier switching graph.

- The upper part made of axioms appearing in a proof-structure is called the *vehicle*. It holds the computational content of the proof (since cuts ultimately only interact with it).
- The lower part is basically the syntax trees of conclusions. It holds the logical content of the proof (formula/correctness). The Danos-Regnier correctness criterion is obtained by considering switchings on the lower part and checking the properties of acyclicity and connectedness for each. This can be understood as *testing* the vehicle against a *set of test* we call the *format*. This test vehicle/format produces a certification: if all tests pass, we have a *proof-net*. This is Girard's "usine" (factory). Note that testing is symmetric because tests are encoded as constellations as well: a format can also be seen as tested by a vehicle.

We call the translation of switching graphs *ordeals*. They are defined in a very natural way by translating the nodes of the lower part of a proof-structure $\mathcal{S}$ into constellations:

- $A^\star = [-t.\mathtt{addr}_{\mathcal{S}}(A), +f.q_A(x)]$ where A is a conclusion of axiom;
- $(A \parr_L B)^\star = [-f.q_A(x)] + [-f.q_B(x), +f.q_{A \parr B}(x)]$;
- $(A \parr_R B)^\star = [-f.q_A(x), +f.q_{A \parr B}(x)] + [-f.q_B(x)]$;
- $(A \otimes B)^\star = [-f.q_A(x), -f.q_B(x), +f.q_{A \otimes B}(x)]$;
- We add $[-f.q_A(x), p_A(x)]$ for each conclusion A.

where $-t.\mathtt{addr}_{\mathcal{S}}(A)$ corresponds to the address of A relatively to $\mathcal{S}$. The colour f stands for "format" and t for "typing".

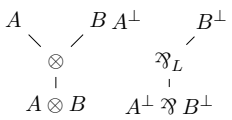
Definition 1 (Logical correctness)**.** *The translation of a proof-structure of conclusion* $\vdash A_1, ..., A_n$ *is said to be* correct *when for each translation of switching graph, their union normalises into* $[p_{A_1}(x), ..., p_{A_n}(x)]$.

However, an important point to notice is that if we have an incorrect closed (because of a cut, e.g $\vdash Cut(A^\perp \parr B^\perp, A \otimes B)$) connected component next to a correct one, the incorrect one becomes invisible because it normalises into $\emptyset$. In order to handle that, we

can either use a notion of *wager* as with Seiller's interaction graphs [34] or accept the empty star which will corresponds to the result of evaluating closed diagrams.

The core point of the translation is that the corresponding dependency graph, obtained by describing how the rays can be connected to each other, will have the same shape as a switching graph.

Example 8. *Here is an example with the switching graph and the test of figure 1:*

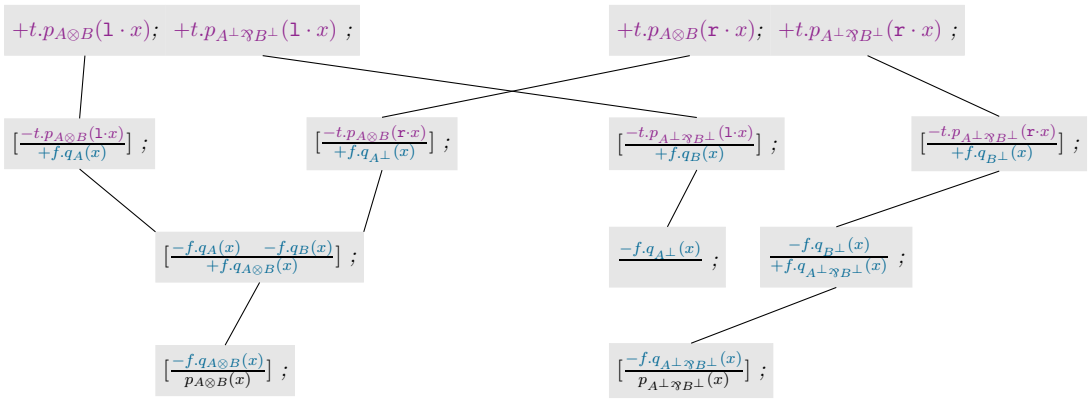


$$[\frac{-t.p_{A\otimes B}(\mathtt{l}\cdot x)}{+f.q_A(x)}] + [\frac{-t.p_{A^\perp \parr B^\perp}(\mathtt{l}\cdot x)}{+f.q_B(x)}] + [\frac{-t.p_{A\otimes B}(\mathtt{r}\cdot x)}{+f.q_{A^\perp}(x)}] + [\frac{-t.p_{A^\perp \parr B^\perp}(\mathtt{r}\cdot x)}{+f.q_{B^\perp}(x)}] + [\frac{-f.q_A(x) \quad -f.q_B(x)}{+f.q_{A\otimes B}(x)}] + [\frac{-f.q_{A^\perp}(x)}{}] + [\frac{-f.q_{B^\perp}(x)}{+f.q_{A^\perp \parr B^\perp}(x)}] + [\frac{-f.q_{A\otimes B}(x)}{p_{A\otimes B}(x)}] + [\frac{-f.q_{A^\perp \parr B^\perp}(x)}{p_{A^\perp \parr B^\perp}(x)}]$$

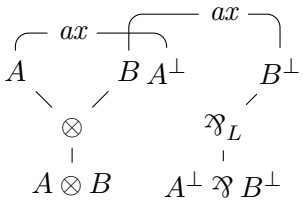
When connected to the vehicle

$$[+t.p_{A\otimes B}(\mathtt{l}\cdot x), +t.p_{A^\perp \parr B^\perp}(\mathtt{l}\cdot x)] + [+t.p_{A\otimes B}(\mathtt{r}\cdot x), +t.p_{A^\perp \parr B^\perp}(\mathtt{r}\cdot x)]$$

we obtain the following dependency graph:

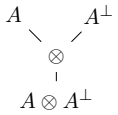


structurally corresponding to the following switching graph:



Since the matching of the constellation is exact and that the dependency graph is a tree, only the free rays will be kept in the normal form. We obtain $[p_{A\otimes B}(x), p_{A^\perp \parr B^\perp}(x)]$.

Example 9. *If we have the following test instead:*



$$[\frac{-t.p_{A\otimes A^\perp}(\mathtt{l}\cdot x)}{+f.q_A(x)}] + [\frac{-t.p_{A\otimes A^\perp}(\mathtt{r}\cdot x)}{+f.q_{A^\perp}(x)}] + [\frac{-f.q_A(x) \quad -f.q_{A^\perp}(x)}{+f.q_{A\otimes A^\perp}(x)}] + [\frac{-f.q_{A\otimes A^\perp}(x)}{p_{A\otimes A^\perp}(x)}]$$

When connected to the vehicle $[+t.p_{A\otimes A^\perp}(\mathtt{l}\cdot x), +t.p_{A\otimes A^\perp}(\mathtt{r}\cdot x)]$*, a loop appears in the dependency graph and since the matching is exact, we can construct infinitely many correct diagrams. The constellation isn't strongly normalising.*

5.3 What is a proof?

We started from very general untyped objects, the constellation and reconstructed the elementary bricks of multiplicative linear logic. Our framework is general enough to write a lot of things which were impossible to write with proof-structures:

- an atomic pre-proof of conclusion $\vdash A$ as the constellation $\{[p_A(x)]\}$,
- an n-ary axiom as the constellation $\{[p_{A_1}(x), ..., p_{A_n}(x)]\}$.

We can finally define what the full translation of a proof-structure is. A proof-structure is translated into a triple (Φ_V, Φ_C, Φ_F) where:

- Φ_V is the uncoloured vehicle made of translations of axioms,
- Φ_C is the uncoloured translation of cuts,
- Φ_F is coloured translation of all ordeals (the format).

We see that proof-structures can be considered as being made of three components. We will then colour the components depending on if we want to perform a cut-elimination or test the correctness. This shows that proof-structures, although being considered untyped, actually come with a kind of pre-made typing. We made the implicit logical assumptions computationally explicit.

6 Two notions of types/formulas

Depending of whether types or programs come first, we have two distinct notions of typing which are reunited in the Transcendental Syntax. Girard's talk about existentialism when programs comes first and essentialism when programs come first.

We first have to define an orthogonality relation $\perp$ opposing constellations depending of what we consider a correct interaction. This is a subjective side a logic. Several choices are possible and lead to different results. For instance, we may consider:

- $\Phi \perp \Phi'$ when $|\mathtt{Ex}(\Phi \uplus \Phi')| < \infty$ (strong normalisation) which captures MLL+MIX correctness.
- $\Phi \perp \Phi'$ when $|\mathtt{Ex}(\Phi \uplus \Phi')| = 1$ which captures MLL correctness.

6.1 A priori typing / à la Church / Girard's factory

In usual proof theory and traditional type theory (for instance Martin-Löf type theory), types come first and can be see as simple labels. We usually want type checking to be tractable by using rules and few good properties ensuring a sound computational behaviour. However, some programs may not be typable (for instance $\lambda x.xx$ in simply typed lambda-calculus). Types represents arbitrary constraints on computation.

With proof-nets, we have proof-structures as general computational objects. We can check if they are correct. What we mean by correct is that it corresponds to a proof of sequent calculus. Here, the Danos-Regnier switchings corresponds to a constraint certifying proof-nets. They are kind of rules telling if a proof-structure is a proof of its conclusions.

The Transcendental Syntax takes inspiration from that and consider a notion of typing by test. We can define an arbitrary set of tests (actually constellation) $\Phi_1, ..., \Phi_n$ and say that a constellation orthogonal to all these tests are of type A. Each formula is then associated to a set of tests.

For instance, we can consider the type Nat for natural numbers and say that $\Phi \bot \Phi'$ when $|\mathtt{Ex}(\Phi \uplus \Phi')| = 1$. We compare the traditional approach and the approach of the Transcendental Syntax where we only have one test.

	Type Theory	Transcendental Syntax
Type checking	$\dfrac{}{0 : \mathrm{Nat}} \quad \dfrac{n : \mathrm{Nat}}{s(n) : \mathrm{Nat}}$	$\Phi_{\mathbf{N}} := [+nat(0), ok] + [-nat(n), +nat(s(n))]$
Correct	$\dfrac{\dfrac{}{0 : \mathrm{Nat}}}{s(0) : \mathrm{Nat}}$	$\mathtt{Ex}(\Phi_{\mathbf{N}} + [-nat(s(0))]) = [ok]$
Incorrect	$0(0)$ not typable	$\mathtt{Ex}(\Phi_{\mathbf{N}} + [-nat(0(0))]) = \emptyset$

6.2 A posteriori typing / à la Curry / Girard's use

Another point of view is to consider realisability techniques [29, 27] for linear logic similarly to ludics [17]. Instead of types as labels, we consider types as groups of programs corresponding to a particular computational behaviour. Types are seen description of behaviour and comes after. It is a notion more general than usual typing. In particular, tests for a formula A are included in the corresponding behaviour $\mathbf{A}$.

Pre-conduct. A pre-conduct is a set of constellations.

Orthogonal. If we have a set of constellations $\mathbf{A}$, its orthogonal, written $\mathbf{A}^\perp$, is the set of all constellations which are strongly normalising when interacting with the constellations of $\mathbf{A}$. It corresponds to linear negation.

Conduct. A *conduct* (or formula) $\mathbf{A}$ is the orthogonal of another pre-conduct $\mathbf{B}$ i.e $\mathbf{A} = \mathbf{B}^{\perp}$. It means that it interacts well (with respects to the orthogonality) with another pre-conduct. It is equivalent to say that $\mathbf{A} = \mathbf{A}^{\perp\perp}$ meaning that it is closed by interaction.

Atoms. We define atoms with a *basis of interpretation* Φ associating for each type variable X_i a distinct conduct $\Phi(X_i)$. It represents a choice of formula for each variable. A more satisfactory way to handle variables is to consider second order quantification, in which case we need further correctness tests. Since our atoms are represented by rays (thus concrete entities), Girard even consider a constants ㇷ (fu) [21] which is self-dual.

Tensor The tensor $\mathbf{A} \otimes \mathbf{B}$ of two conducts is constructed by pairing all the constellations of $\mathbf{A}$ with the ones of $\mathbf{B}$ by using a multiset union of constellations $\Phi_1 \uplus \Phi_2$. The conduct $\mathbf{A}$ and $\mathbf{B}$ have to be disjoint in the sense that they cannot be connected together by two matching rays. Note that the cut is the same thing but the constellations can interact.

Par and linear implication As usual in linear logic, the par and linear implication are defined from the tensor and the orthogonal: $A \parr B = (A^{\perp} \otimes B^{\perp})^{\perp}$ and $A \multimap B = A^{\perp} \parr B$.

Alternative definition for linear implication An alternative but equivalent definition of the linear implication $\mathbf{A} \multimap \mathbf{B}$ is the set of all constellations Φ such that if we put them together with any constellation of $\mathbf{A}$, the execution produces a constellation of $\mathbf{B}$.

Example 10 (acceptation in finite automata)**.** *From the previous encoding of automata we can observe a duality between automata and words. It induces an orthogonality:* $A^{\bigstar} \perp w^{\bigstar}$ *when* $[accept] \in \mathtt{Ex}(A^{\bigstar} + w^{\bigstar})$*. An automaton becomes orthogonal to all the words it accepts and a word is orthogonal to all the automata which recognise it.*

Example 11 (queries and answers in logic programming)**.** *We can sketch idea of typing for logic programming. Let*

$$\Phi^{+}_{\mathbf{N}} = [+add(0, y, y)] + [-add(x, y, z), +add(s(x), y, s(z))]$$

be a constellation. We consider the strong normalisation of the union of two constellations as orthogonality. Let $\mathbf{QAdd} = \{[-add(s^n(0), s^m(0), r), r] \mid n, m \in \mathbf{N}\}$ *and* $\mathbf{AAdd} = \{[s^n(0)] \mid n \in \mathbf{N}\}$*. Take a constellation* $[-add(s^n(0), s^m(0), r), r]$ *and connect it with* $\Phi^{+}_{\mathbf{N}}$*. All diagrams corresponding to* $\Phi^{+}_{\mathbf{N}}$ *with* n *occurrences of*

$$[+add(s(x), y, s(z)), -add(x, y, z)]$$

can be reduced to a star $[s^{n+m}(0)]$*. It is easy to check that all other diagram fails. Therefore, for all* $\Phi \in \mathbf{QAdd}$*,* $\mathtt{Ex}(\Phi \uplus \Phi^{+}_{\mathbf{N}}) \in \mathbf{AAdd}$ *and* $\Phi^{+}_{\mathbf{N}} \uplus \Phi$*. If* $\mathbf{QAdd}$ *and* $\mathbf{AAdd}$ *are proven to be conducts (we need a more specific orthogonality) then* $\Phi^{+}_{\mathbf{N}} \in \mathbf{QAdd} \multimap \mathbf{AAdd}$*. Although not explored here, it might be possible to retrieve existing type systems and extend them by sound constructions.*

We can imagine more interesting examples: typing for tilings models (thus typing for DNA computing), characterisation of properties of constellations (characterisation of complexity classes?).

As in the theory of classical realisability, we can consider an adequacy lemma linking the two notions of typing. It states that a finite set of tests for formula A is sufficient in order to ensure membership to the conduct $\mathbf{A}$ corresponding to A.

Definition 2 (Adequacy property)**.** *Let Φ be a constellation and $\Phi_1, ..., \Phi_n$ be tests for a formula A. If for all $1 \leq i \leq n$, $\Phi \perp \Phi_i$ then $\Phi \in \mathbf{A}$ where $\mathbf{A}$ is the conduct corresponding to A (it is possible to translate a formula into a conduct by substituting the variables by other conducts).*

It the case of MLL, it makes explicit the fact that the correction graphs corresponds to sort of pre-proofs of $A^\perp$ for a proof-structure of conclusion A because the constellations $\Phi_1, ..., \Phi_n$ corresponding to tests for a formula A form a pre-conduct included in $\mathbf{A}^\perp$.

6.3 A logical constant from a topological invariant

In Transcendental Syntax's fourth paper [22], Girard provide hints for the definition of a self-dual conduct corresponding to a new logical constant.

The idea is that in the stellar resolution, the atoms are concrete objects and not substituable ones like in the original theory of proof-nets. It should be possible to group them in a conduct corresponding to the type of atoms, analogous to the conduct containing the partition $\{\{1\}\}$. But the only thing they share in common is their topology (a single point). In order to discover Girard's logical constant $\curlywedge$ we consider a new point of view on logical correctness based on a topological invariant: the Euler-Poincaré invariant.

Definition 3 (Euler-poincaré invariant)**.** *For a convex polyhedron surface, we define the Euler-Poincaré invariant by $|Vertices| - |Edges| + |Faces| = 2$.*

It can be adapted to graphs where the number of faces corresponds to the number of regions. For graphs we have $|Vertices| - |Edge| = 1$. In the case of partitions, when we see them as bipartite graphs we have two partitions P, Q on $\{1, ..., N\}$ interacting. We have $|P| + |Q| - N = 1$. We can multiply by 2 in order to get $(2|P| - N) + (2|Q| - N) = 2$ exhibiting a kind of independent weight function $\omega(P) = 2|P| - N$ for the partitions. It gives us a reformulation of Danos-Regnier criterion. For each vehicle P and test Q, we should have $\omega(P) + \omega(Q) = 2$. We can formulate this in constellations as an orthogonality relation only taking the topology of constellations into account.

Definition 4 (Weight)**.** *The* weight *of a constellation Φ is defined by $\omega(\Phi) = 2|\Phi| - |\mathtt{Rays}(\Phi)|$ where $\mathtt{Rays}(\Phi)$ is the set of all rays occurring in Φ.*

Definition 5 (Orthogonality)**.** *Two constellations Φ, Φ' are* orthogonal*, written $\Phi \perp \Phi'$, if and only if $\omega(\Phi) + \omega(\Phi') = 2$ and $|\mathtt{Rays}(\Phi)| = |\mathtt{Rays}(\Phi')|$.*

We can then define a conduct corresponding to the type of atoms.

Proposition 1 (Logical constant フ)**.** *The pre-conduct* $\text{フ} = \{\{\phi\} \mid |\phi| = 1\}$ *is a self-dual conduct.*

Now, we can give ground to multiplicative propositions. All type variables can be replaced by フ. For instance $\vdash A^{\perp} \parr B^{\perp}, A \otimes B$ becomes $\vdash \text{フ} \parr \text{フ}, \text{フ} \otimes \text{フ}$.

Remark few interesting points:

- for any formula A, we can type atomic pre-proof: $[p_A(x)] \in \text{フ}$;
- the axioms have a type: $[p_A(x), p_{A^{\perp}}(x)] \in \text{フ} \parr \text{フ}$;
- we can write stand-alone links: $[p_A(x), p_B(x)] \in \text{フ} \parr \text{フ}$ and $[p_A(x)] + [p_B(x)] \in \text{フ} \otimes \text{フ}$.

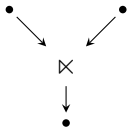
7 Extension to intuitionistic exponentials

The exponentials have already been studied using flows (corresponding to binary stars) [16, 7] but the correctness has not been taken into account yet.

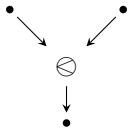
The idea is that the stellar resolution already have a built-in mechanism of duplication (by multiple matching and reusing of stars) and weakening (destruction of some unwanted diagrams).

We restrict the definitions to intuitionistic exponentials in order to follow Girard's treatment of full exponentials in second order.

- Non-linear formulas are written $\underline{A}$ (this corresponds to $?A$).
- We define new connectives $A \ltimes B$ ($=?A \parr B$) and $A \circledcirc B$ ($=!A \otimes B$) with the following links:



Exponential par



Exponential tensor

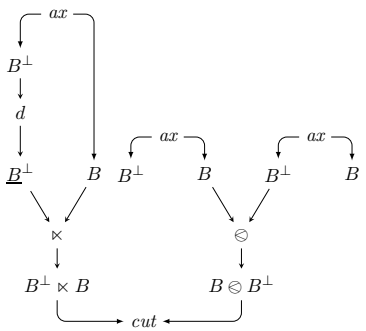
- Weakened formulas are not translated.
- Derelicted formulas $\underline{A}$ are translated as $p_A(t \cdot \mathtt{d})$.
- Contracted formulas $\underline{A}$ are translated into $p_A(t \cdot (\mathtt{l} \cdot y))$ and $p_A(t \cdot (\mathtt{r} \cdot y))$.
- Promoted formulas A are translated into $p_A(t \cdot y)$ for a fresh variable y and its auxiliary formulas B_i into $p_{B_i}(t_i \cdot y)$.

We can remark that subterms of the form $t \cdot u$ are used. The term t corresponds to the multiplicative part encoding the address of atoms as before and the term u encodes the nested boxes (e.g $(t \cdot y_1) \cdot y_2$) and the copy identifier for the contraction. In the case of dereliction, we prevent any duplication on the current layer by a constant $\mathtt{d}$. This gives a box-free approach to exponentials where box dependency is simulated by matchable terms.

7.1 The computational content of exponentials

We illustrate the cut-elimination by few examples by using the usual translation of simply typed lambda-terms into proof-nets [9, 33].

Example 12 (Case dereliction/box)**.** *This corresponds to opening a box. We illustrate this case with the identity function applied to an argument:* $(\lambda x.x)y$.

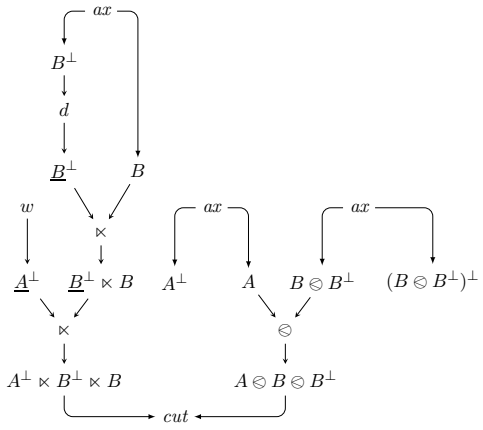


It is translated into:

$$[+c.p_{B^\perp \parr B}((\mathtt{l} \cdot x) \cdot \mathtt{d}), +c.p_{B^\perp \parr B}(\mathtt{r} \cdot x)]+$$
$$[p_{B^\perp}(x), +c.p_{B \otimes B^\perp}((\mathtt{l} \cdot x) \cdot y)]+$$
$$[+c.p_{B \otimes B^\perp}(\mathtt{r} \cdot x), p_B(x)]+$$
$$[-c.p_{B^\perp \parr B}(x), -c.p_{B \otimes B^\perp}(x)]$$

It is similar to the multiplicative case. The essential point is that $+c.p_{B^\perp \parr B}((\mathtt{l} \cdot x) \cdot \mathtt{d})$ *will interact with* $+c.p_{B \otimes B^\perp}((\mathtt{l} \cdot x) \cdot y)$ *replacing* y *by* $\mathtt{d}$*. It means that the rays corresponding to this box are no longer duplicable. In some sense, we opened the box.*

Example 13 (Case weakening/box)**.** *It corresponds to a box erasure. We consider the simple example of right projection* $(\lambda xy.y)z$.



It is translated into the constellation:

$$[+c.p_{A^\perp \parr B^\perp \parr B}((\mathtt{r} \cdot \mathtt{l} \cdot x) \cdot \mathtt{d}),$$
$$+c.p_{A^\perp \parr B^\perp \parr B}(\mathtt{r} \cdot \mathtt{r} \cdot x)]+$$
$$[p_{A^\perp}(x \cdot y), +c.p_{A \otimes B \otimes B^\perp}((\mathtt{l} \cdot x) \cdot y)]+$$
$$[+c.p_{A \otimes B \otimes B^\perp}(\mathtt{r} \cdot x), p_{A \otimes B \otimes B^\perp}(x)]+$$
$$[-c.p_{A^\perp \parr B^\perp \parr B}(x), -c.p_{A \otimes B \otimes B^\perp}(x)]$$

We can see that the cut will be duplicated into $[-c.p_{A^\perp \parr B^\perp \parr B}((\mathtt{l} \cdot x) \cdot y)$, $-c.p_{A \otimes B \otimes B^\perp}((\mathtt{l} \cdot x) \cdot y)]$ *and* $[-c.p_{A^\perp \parr B^\perp \parr B}(\mathtt{r} \cdot x), -c.p_{A \otimes B \otimes B^\perp}(\mathtt{r} \cdot x)]$.

Since $-c.p_{A^\perp \nltimes B^\perp \nltimes B}((\mathtt{l}\cdot x)\cdot y)$ *has nothing to match (because of the weakening), all diagrams using terms matchable with* $-c.p_{A\otimes B\otimes B^\perp}((\mathtt{l}\cdot x)\cdot y)$ *will be excluded. This indeed corresponds to box erasure. Notice that if we had auxiliary gates, they would end up as independent stars in the normal form, as usual with proof-nets.*

Example 14 (Case contraction/box)**.** *It corresponds to a box duplication. Instead of illustrating it by the translation of a lambda-term, we will describe how it works:*

$$[+c.p_A(t\cdot(\mathtt{l}\cdot y)),\ldots]+[+c.p_A(t\cdot(\mathtt{r}\cdot y)),\ldots]+[+c.p_B(t\cdot y),\ldots]+[-c.p_A(x),-c.p_B(x)]$$

The cut star forces the matching between rays for A *and for* B*. We see that* $+c.p_B(t\cdot y)$ *can interact with both* $+c.p_A(t\cdot(\mathtt{l}\cdot y))$ *and* $+c.p_A(t\cdot(\mathtt{r}\cdot y))$*, hence the star containing* $+c.p_B(t\cdot y)$ *and all the stars connected to it will be naturally duplicated.*

7.2 The logical content of exponentials

As usual with proof-nets, if we allow the MIX rule then the correctness criterion is straightforward. In our case, we only need to check acyclicity. The only difference with the multiplicative case is that we should check that the variables we use are handled correctly:

- in subterms $x\cdot y$, the variables x and y must be different;
- the left part of a $\nltimes$ must be of the shape $x\cdot y$.

If we don't allow the MIX rule, only the left part of $\nltimes$ which can be cancelled is problematic. We present the tests for IMELL (intuitionistic MELL) by translating the hyperedges of the lower part of the proof-structure:

- $A^\star=[-t.\mathtt{addr}_{\mathcal{S}}(A),+f.q_A(x\cdot y)]$ where A comes from the left part of $\nltimes$ or $\otimes$;
- $(A\otimes_x B)^\star=[-f.q_A(x\cdot x),-f.q_B(x),+f.q_{A\otimes B}(x)]$;
- $(A\otimes_{\mathtt{1}} B)^\star=[-f.q_A(x\cdot\mathtt{1}),-f.q_B(x),+f.q_{A\otimes B}(x)]$;
- $(A\nltimes_L B)^\star=[-f.q_A(x\cdot y),+f.q_{A\nltimes B}(x\cdot y)]$ (cancelling);
- $(A\nltimes_R B)^\star=[-f.q_B(x),+f.q_{A\nltimes B}(x)]+[-f.q_A(x\cdot y)]+[-f.q_A(x'\cdot y')]$ (unless $x=x'$);
- $(\underline{A})^\star=[-f.q_A(x)]$ where $\underline{A}$ is a conclusion.

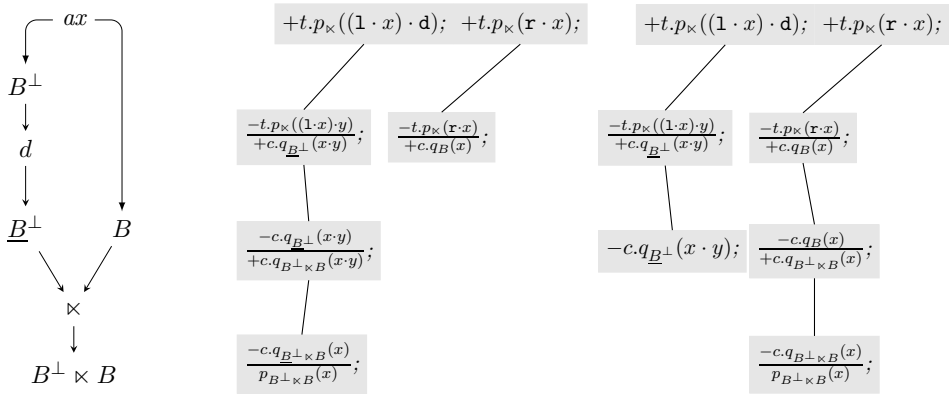
The two tests for $\otimes$ force the presence of different variables. The test $\nltimes_R$ only cancels the non-linear formulas. The star $[-f.q_A(x'\cdot y')]$ is only used when x' can be instanciated with something different from the variable x. This is a technical hack which force the t in the $t\cdot u$ cancelled to be x.

Let us focus on the test $\nltimes_L$ which relies on the mechanisms of stellar resolution. Structural rules only appear on the left of $\nltimes$ and we need both acyclicity and connectedness. Remark that we ask for this test to be cancelling (the interaction normalises into the empty constellation $\emptyset$). The test connects the conclusion of $\nltimes$ to its left part. The possible cases for $A\nltimes B$ are the following ones:

- the stars connected to A form an independent connected component and form an open diagram and the normal form is non-empty. If the corresponding diagram is closed then the empty star [] appears in the normal form.
- a copy of A is connected to $A \ltimes B$ by the bottom (with a cut). In this case, we have a cycle and infinitely many diagrams. The constellation is not strongly normalising.
- the left part of $\ltimes$ is connected to the right part either from the top. The whole constellation only normalises into $\emptyset$ when every formulas are connected. Cycles always involve exact matching, hence we must have acyclicity of the dependency graph.

We then ask that for each tests (except the ones containing $\ltimes_L$ which should be cancelled) we get the star of conclusions but we also require that we only get the non-linear conclusions. Hence, for a proof of $\vdash \Gamma, \underline{\Delta}$, we should obtain the star of conclusions of Γ. This is due to a recurrent problem (not necessarily a flaw) of GoI: it is impossible to act on the auxiliary conclusions of boxes and to always preserve the paths of proofs with non-linear conclusions. Since we consider IMELL instead of MELL, non-linear conclusions $\underline{A}$ cannot appear as conclusion of a proof anyway.

Example 15 (Correctness of identity function)**.** *We check the correctness of the lambda-term $\lambda x.x$.*



We obtain two diagrams corresponding to correction graphs. The left one using $\ltimes_L$ is removed because $+c.q_B(x)$ has nothing to match. The right one using $\ltimes_R$ normalises into $[p_{B^\perp\ltimes B}(x)]$. Hence, the identity function is logically correct.

8 New horizons for second order logic

A novelty of the Transcendental Syntax is the computational reconstruction of predicate calculus but also a new point of view on second order. We first explain Girard's distinction between first and second order logic which is different from their original meaning.

8.1 Girard's first and second order

Girard explained in a paper written in French [21] that second order logic is actually implicit in a lot of definitions of logic. For instance, the rule $(\vee e)$ for NJ implicitly requires a generic formula C:

$$\dfrac{\begin{array}{ccc} & [A] & [B] \\ \vdots & \vdots & \vdots \\ A \vee B & C & C \end{array}}{C} \vee e \qquad\qquad \dfrac{A \quad B}{A \wedge B} \wedge i$$

In theorems of propositional logic such as $A \Rightarrow B \Rightarrow A$, the formulas A and B are implicitly universally quantified and should be understood as $\forall A\ B.\ A \Rightarrow B \Rightarrow A$. The same thing occurs for P which is external in $\forall x.P(x)$. This is how Girard justifies the treatment of the additives and exponentials in second order for proof-nets.

If we look at the rule $\wedge i$, it only reunites hypotheses without any genericity. Since MLL proof-nets are hypergraphs linking formulas, it is similar. Even the atoms are concrete objects: simple vertices with non-essential labels. Only the location truly matters.

First-order logic corresponds to the part of logic which uses nothing more than "what is already there" and second order logic as the part of logic referring to the set of all propositions. Therefore, predicate calculus is actually part of second order logic while Intuitionistic MELL with ⅂ is purely first-order. We have the new following distinction between first and second order:

First-order $\otimes, \parr, \multimap, ⅂, ∀, \ltimes, \rtimes, \Rightarrow$;

Second order $?, !, \oplus, \&, \forall, \exists, \mathbf{1} = !\top, \bot = ?\mathbf{0}, \mathbf{0} = \forall X.X, \top = \exists X.X$.

where the additives (not detailled here) are defined as following:

$$A \& B := \exists X.(!(X \multimap A) \otimes !(X \multimap B) \otimes X)$$

$$A \oplus B := \forall X.((A \multimap X) \Rightarrow ((B \multimap X) \Rightarrow X)).$$

These new formulations for additives comes from consideration of the rules for $\wedge$ and $\vee$ in **NJ**.

8.2 Girard's derealism

Girard's *derealism* corresponds to a new status for proof coming from a new treatment of second order quantification. His intuition comes from the second order definition of natural numbers:

$$\mathbf{nat} := \forall X.(X \multimap X) \Rightarrow (X \multimap X).$$

He claims [21] that $\mathbf{nat}^{\perp}$ corresponds to iteration/induction on natural numbers and that if testing for $\mathbf{nat}^{\perp}$ was finite, we could determine which iterations are licit, which is problematic. Further details are needed in order to make this idea explicit but it is not investigated here. Girard has the intuition that reasoning should be finite, hence we

should preserve a finite testing. This leads to quantified entities being part of the proof itself (T being part of a proof of $\exists X.A$ coming from $A[X := T]$).

A proof is now a tuple $(\Phi_V \uplus \Phi_M, \Phi_C, \Phi_F)$ where Φ_M is called the *mould* or *witness* of the proof and corresponds to a test associated to witnesses (the T associated to X in $\exists X.A$).

Since X in $\exists X.A$ may appear positively or negatively (X or $X^\perp$), the witness comes in two versions and we are interested in an equilibrum between them which is not discussed here but the reader can find more details in the fourth article of Transcendental Syntax [22].

For technical reasons and to add more combinatorial complexity to proofs, we distinguish two classes of rays:

- *objective* rays which are the usual rays;
- *subjetive* rays which are rays which have internal colours, for instance $+a(-b(x))$. As a consequence, we have to consider a subjective logical constant. Girard calls it $\Finv$ (wo). It can be tested with the test $[-t.p_{\Finv}(t \cdot x), p_{\Finv}(x)]$ where t is subjective (e.g $+a(x)$) ensuring the presence of an internal colour.

A star is objective if it only has objective rays and subjective if it only has subjective rays. Otherwise, it is *animist* (a mix of objective and subjective). An *épure* is a constellation without animist star, i.e. it can be put into the form $\Phi_O \uplus \Phi_S$ where Φ_O only contains objective stars and Φ_S subjective ones. This makes the idea of correct proof clearer: it has to be made of an épure with an objective part only containing binary stars (they represent axioms).

The interesting feature of this new combinatorics is that the status of stars (objective/subjective/animist) can change during the execution. Another consequence is that we can design a conduct $\mathbf{0}$ containing a constellation which can interact with other constellations but which is not correct (because of animist stars). This gives this constant a computational content.

8.3 Additive neutrals

We illustrate the mechanisms of witnesses and subjective rays with additive neutrals by following the third paper of Transcendental Syntax [20]. We define the following rays:

$$C_\top(x) := p_\top(\mathtt{c} \cdot x) \qquad L_\top(x) := p_\top(-t(\mathtt{l} \cdot x)) \qquad R_\top(x) := p_\top(-t(\mathtt{r} \cdot x))$$

The neutral element $\top$ is defined by the orthogonal of the following tests where the second one is cancelling (as for $\ltimes_L$ in the exponentials):

$$\top_1 : \left[\frac{-t.C_\top(x), -t.L_\top(x)}{}\right] + \left[\frac{-t.R_\top(x)}{p_\top(x)}\right] \qquad \top_2 : \left[\frac{-t.C_\top(x), -t.L_\top(x)}{p_\top(x)}\right] \quad \text{(cancel)}$$

The constellation $[+t.C_\top(x)] + [+t.L_\top(x), +t.R_\top(x)]$ is an épure passing the tests. However, it is not a correct proof because the objective part is unary and not binary.

If we take $[+t.C_\top(\mathtt{l}\cdot x), +t.C_\top(\mathtt{r}\cdot x)] + [+t.L_\top(\mathtt{l}\cdot x)] + [+t.L_\top(\mathtt{r}\cdot x), +t.R_\top(x)]$ instead, then it is a correct and also passes the test. In fact, it is possible to check that for any $\Phi \in \top$, we have $\Phi \in ?(\mathcal{フ} \otimes \mathcal{ヲ}) \parr \mathcal{ヲ} = (\mathcal{フ} \otimes \mathcal{ヲ}) \ltimes \mathcal{ヲ} = (\mathcal{フ} \parr \mathcal{ヲ}) \Rightarrow \mathcal{ヲ}$ because of the possibility of duplicating $C_\top$ and $L_\top$ and still pass the tests.

We define the following rays:

$$C_{\mathbf{0}}(x) := p_{\mathbf{0}}(\mathtt{c}\cdot x) \qquad L_{\mathbf{0}}(x) := p_{\mathbf{0}}(-t(\mathtt{l}\cdot x)) \qquad R_{\mathbf{0}}(x) := p_{\mathbf{0}}(-t(\mathtt{r}\cdot x))$$

The neutral element $\mathbf{0}$ is defined from the following tests:

$$\mathbf{0}_1 : \left[\frac{-t.C_{\mathbf{0}}(x)}{}\right] + \left[\frac{-t.L_{\mathbf{0}}(x), -t.R_{\mathbf{0}}(x)}{p_{\mathbf{0}}(x)}\right] \qquad \mathbf{0}_2 : \left[\frac{-t.L_{\mathbf{0}}(x)}{}\right] + \left[\frac{-t.C_{\mathbf{0}}(x), -t.R_{\mathbf{0}}(x)}{p_{\mathbf{0}}(x)}\right]$$

$$\mathbf{0}_3 : \left[\frac{-t.R_{\mathbf{0}}(x)}{p_{\mathbf{0}}(x)}\right]$$

The constellation $[+t.R_{\mathbf{0}}(x)]+[+t.C_{\mathbf{0}}(x), +t.L_{\mathbf{0}}(x)]$ (containing an animist star) passes the tests but there is no épure. The reason is that the tests are designed so that the orthogonal of the tests must have a mix of objective and subjective rays. This new combinatorics for proofs allows to speak about logical coherence. Since $\mathbf{0} = \top^\perp$, we should have that for any $\Phi \in \mathbf{0}$, we have $\Phi \in (\mathcal{フ} \parr \mathcal{ヲ}) \oslash \mathcal{ヲ}$ (by following the expression of $\top$ in terms of $\mathcal{フ}$ and $\mathcal{ヲ}$).

8.4 Predicate calculus

Because of Girard's new distinction between first and second order, I choose to use the term "Predicate calculus" instead of "First-order logic" to avoid confusion.

As remarked before, the predicate calculus is part of second order logic but few conceptual choices remain. Girard remarks that by looking at Leibniz's equality defining $a = b$ as $\forall X.X(a) \Leftrightarrow X(b)$, the predicate X would play no role if written as a proof-net. We can connect a and b directly and X is seen as a sort of modality restricting connexions instead. This leads to the idea of representing terms/individuals as multiplicative formulas and equality as linear equivalence $A \equiv B$ defined as $(A \multimap B)\&(B \multimap A)$. In terms of derealism, the witnesses encode formats corresponding to multiplicative formulas. The encodings are free but for individuals, we need to ensure technical properties such as the injectivity of the encoding in order to have that $f(x) = f(y)$ implies $x = y$.

We do not give intuitive details about how it is treated in the Transcendental for the moment and refer to Girard's third article [20].

9 Beyond logic

What follows is a possibly exaggerated and naive discussion about how logic could actually be more than "our" logic. Something I find quite remarkable is that the stellar resolution places the link between computation and logic in the broad world of emergence

and complex systems. We compute by non-deterministic local interactions between independent entities with a notion of information propagation. Logic emerges from the behaviour of constellations. Since it is close to tiling models, we can also expect relationships with sequential dynamical systems and graph dynamical systems.

At the time of the document, I believe than no one understands the nature of logic. The Geometry of Interaction and the Transcendental Syntax showed that we can speak about linear logic with simple topology/geometry/dynamics by considering the locations of entities, their links, cycles, interaction, paths etc (a very general setting exists in Seiller's works [38]). Rather tangible things reminiscent of interaction in the biological/physical world (for instance chemical reaction networks [26]).

I wonder if it is possible to consider a setting even more general than stellar resolution, for instance a dynamical system for which attractors would have a logical meaning. It may be possible that what we know about logic is only a particular case of something bigger, a kind of *theory of interaction* as mentioned several times already [1, 15].

Indeed, we must take several things into account. First, logic has a subjective side because we look at computation from a specific point of view and a choice of what a sound/successful computation means, which depends on the use of objects and our own perception/motivations. It probably makes purely mathematical techniques probably insufficient (unless we speak about the objective part of logic, i.e. computation). A conceptual work and connexions with other fields (typically physics and biology) may be useful and seems a priori possible. Moreover, we can ask if logic exists beyond the computable.

Finally, I am curious about whether all of that has something to say at all about computational complexity (some hints seem to exists in Seiller's works [39, 36]). Due to the barriers of problems such as the separations of classes, it may be necessary to look for new definitions of logic of computation.

References

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

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

[3] 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.

[4] Andrea Asperti and Cosimo Laneve. Paths, computations and labels in the λ-calculus. *Theoretical Computer Science*, 142(2):277–297, 1995.

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

[6] 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.

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

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

[9] Vincent Danos. *La Logique Linéaire appliquée à l'étude de divers processus de normalisation (principalement du Lambda-calcul)*. PhD thesis, Paris 7, 1990.

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

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

[12] Jean-Yves Girard. Multiplicatives. 1987.

[13] Jean-Yves Girard. Geometry of interaction II: deadlock-free algorithms. In *International Conference on Computer Logic*, pages 76–93. Springer, 1988.

[14] 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.

[15] Jean-Yves Girard. Towards a geometry of interaction. *Contemporary Mathematics*, 92(69-108):6, 1989.

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

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

[18] Jean-Yves Girard. Geometry of interaction IV: the feedback equation. In *Logic Colloquium*, volume 3, pages 76–117, 2006.

[19] Jean-Yves Girard. Geometry of interaction V: logic in the hyperfinite factor. *Theoretical Computer Science*, 412(20):1860–1883, 2011.

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

[21] Jean-Yves Girard. La logique 2.0. 2018.

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

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

[24] 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.

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

[26] Jürgen Jost and Raffaella Mulas. Hypergraph laplace operators for chemical reaction networks. *Advances in mathematics*, 351:870–896, 2019.

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

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

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

[30] Alexander Leitsch. *The resolution calculus*. Springer Science & Business Media, 2012.

[31] Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(2):258–282, 1982.

[32] Matthew J Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 13(2):195–224, 2014.

[33] Laurent Regnier. *Lambda-calcul et réseaux*. PhD thesis, Paris 7, 1992.

[34] Thomas Seiller. Interaction graphs: multiplicatives. *Annals of Pure and Applied Logic*, 163(12):1808–1837, 2012.

[35] Thomas Seiller. *Logique dans le facteur hyperfini: géometrie de l'interaction et complexité*. PhD thesis, Aix-Marseille Université, 2012.

[36] Thomas Seiller. Towards a complexity-through-realisability theory. *arXiv preprint arXiv:1502.01257*, 2015.

[37] 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.

[38] Thomas Seiller. Interaction graphs: Graphings. *Annals of Pure and Applied Logic*, 168(2):278–320, 2017.

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

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

[41] Wolfgang Thomas. On logics, tilings, and automata. In *International Colloquium on Automata, Languages, and Programming*, pages 441–454. Springer, 1991.

[42] Hao Wang. Proving theorems by pattern recognition —II. *Bell system technical journal*, 40(1):1–41, 1961.

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