Stellar Resolution: Multiplicatives - for the linear logician, through examples

Boris Eng

Stellar Resolution: Multiplicatives

For the linear logician, through examples

Boris Eng

The stellar resolution is an asynchronous model of computation used in Girard's Transcendental Syntax [7, 10, 8, 9, 11] which is based of Robinson's resolution [14]. Similarly to classical realisability, we obtain a new model of linear logic with computational objects as its foundation: proofs, cut-elimination, formulas/types, correctness and provability are reconstructed very naturally. In this paper, we investigate the case of the multiplicative fragment of linear logic.

1 Stellar Resolution

1.1 Frequently Asked Questions

- *What is the Transcendental Syntax about?* The Transcendental Syntax is a programme initiated by Girard which can be seen as the successor of his Geometry of Interaction (GoI) programme [3, 2, 1, 4, 5, 6] studying linear logic from the mathematics of the cut-elimination. In the same idea, the Transcendental Syntax aims at giving a fully computational foundation for logic where entities such as formulas, proofs, correctness, truth are reconstructed from scratch with a computational model as it is done in classical realisability for instance. One can understand it as a reverse engineering of logic from its computational behaviours.

- *Where does the model of stellar resolution come from?* We use "stellar" for Girard's terminology and "resolution" for its similarities the resoltion-based models. The GoI began with a mathematical study of the cut-elimination procedure through the use of operator algebras. In the third article of GoI, Girard discovered a simplification based on first-order unification: the model of flows which is basically unary first-order resolution. The stellar resolution is simply an extension of this model. One may choose another model of computation as a basis of the Transcendental Syntax but the stellar resolution is a natural and convenient one.

- *Is it related to any other works?* The stellar resolution is able to simulate models of computation which are also dynamical systems: abstract tile assembly models [13] which are used in DNA computing [17] but also the computational model of Wang

tiles [16]. From our realisability construction, one can imagine methods of typing and implicit complexity analysis of these models. The stellar resolution can also be seen as a generalisation of the model of flows used in GoI and of Seiller's interaction graphs [15]. The reconstruction of types/formulas follow the constructions of the model of realisability.

- *Why is it interesting?* The stellar resolution generalises flows and interaction graphs which have an applications in implicit computational complexity. The Transcendental Syntax programme should be able to produce a more refined notion of type/formula but also, more interestingly, to provide a computational content for first-order logic in the sense of the Curry-Howard correspondence, something which has not be done yet. Such an interpretation of first-order logic may have applications in descriptive complexity.

1.2 Stars and constellations

We define *rays* by the following grammar:

$$r ::= +a(t_1, ..., t_n) \mid -a(t_1, ..., t_n) \mid t$$

where $+/-$ are polarties, a is a function symbol called a *colours* and $t_1, ..., t_n$ are first-order terms.

A *star* is a finite and non-empty multiset of rays $\sigma = [r_1, ..., r_n]$ and a constellation $\Sigma = \sigma_1 + ... + \sigma_m$ 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.

Example 1. *This is basically a reformulation of first-order resolution. We write s^n for n applications of the symbol s (for instance $s^3(0) = s(s(s(0)))$). A colour is a predicate and the polarity represents the distinction input/ouput or hypothesis/conclusion. The absence of polarity means that the predicate is isolated and cannot be connected. Let's take two logic programs and their associated constellation to illustrate the model:*

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

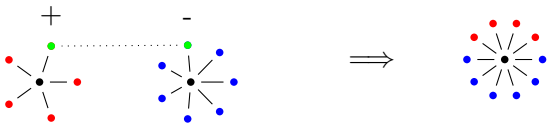
$$\Sigma_{\mathbf{N}}^{n,m} = [+add(0, y, y)] + [+add(s(x), y, s(z)), -add(x, y, z)] + [-add(s^n(0), s^m(0), r), r]$$

```
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).
```

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

More graphically, a star may be depicted as an actual star:  for $[t_1, ..., t_5]$.

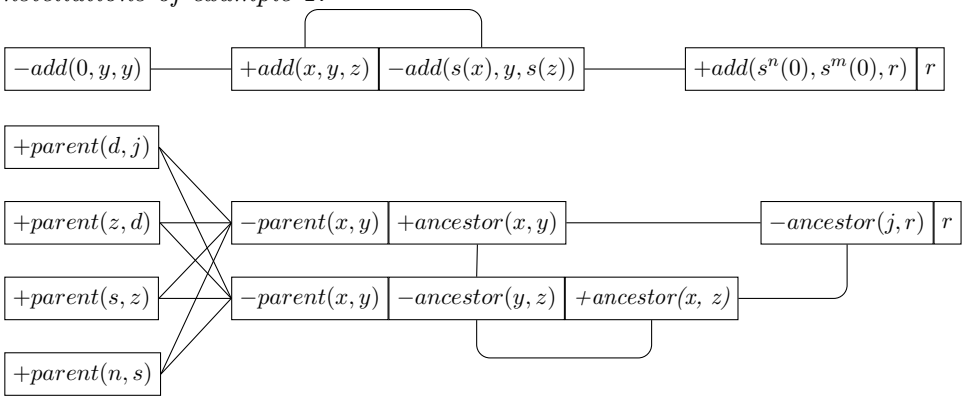
1.3 Evaluation of constellations



Graphically, we evaluate a constellation by connecting rays together when they are matchable and of opposite polarity. The two stars will fuse and the connected rays will disappear. The rays of the remaining star is affected by a "reaction" of this fusion.

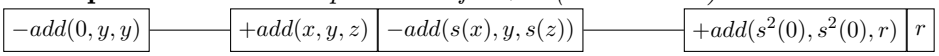
To do so, from a constellation Σ, we can construct its *unification graph* $\Sigma[\mathcal{A}]$ for a set of colours $\mathcal{A}$ telling us which rays can be connected together. It is a graph of stars $\sigma \in \Sigma$ with edges between two stars whenever there are two rays r_1, r_2 of opposite polarity such that their underlying terms (without colour) are matchable. The edge is then labelled with the equation $r_1 \doteq r_2$.

Example 2. *We give a representation of the unification graphs corresponding to the constellations of example 1:*

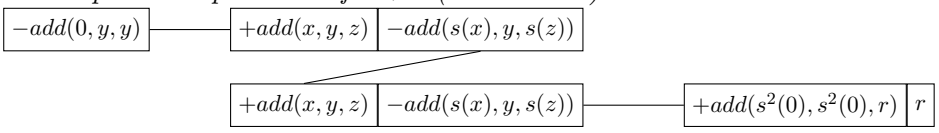


We can then make actual connexions between *occurrences* of stars following the unification graph of a constellation. Such a connexion, called a *diagram* (written δ) is connected and usually a tree (when considering logic) where all star variables have to be made distinct. They represent computations to be done.

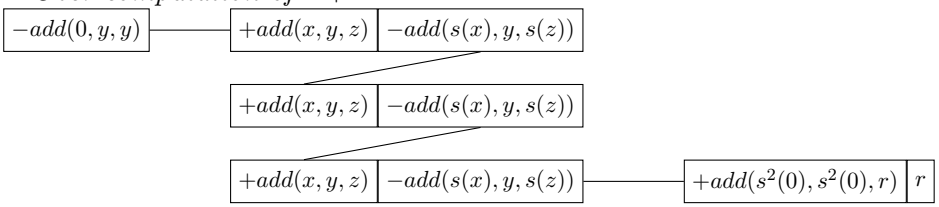
Example 3. *Partial computation of* $2+2$ *(0 recursion):*



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



Over computation of $2+2$*:*

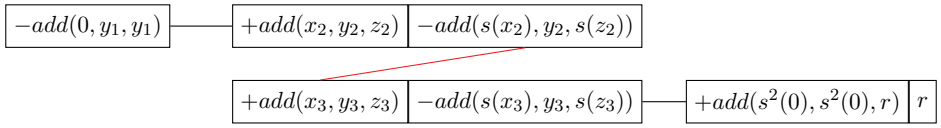


There are two equivalent ways to reduce diagrams by observing that each edge define a unification equation:

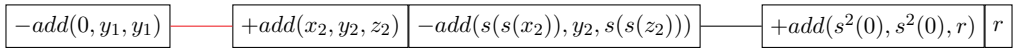
Fusion We can reduce the links step by step by solving the underlying equation, producing a solution θ. The two linked stars will fuse by making the connected rays disappear. The susbstitution θ is finally applied on the rays of the resulting star. This is exactly the resolution rule.

Actualisation The set of all edges define a unification problem. The solution θ of this problem is then applied on the star of free rays (unconnected rays).

Example 4 (fusion)**.** *The full fusion of the diagram representing a complete computation of* $2+2$ *from example 3 is described below (we make the exclusion of variable explicit for illustration):*

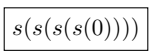


$$\downarrow \theta = \{x_3 \mapsto s(x_2), y_3 \mapsto y_2, z_3 \mapsto s(z_2)\}$$



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

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



Example 5 (actualisation)**.** *If we take the diagram δ representing a complete computation in the example 3, it generates the following problem:*

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

which is solved by a unification algorithm such as the Montanari-Martelli algorithm [12] in order to obtain a finale substitution:

$$\begin{aligned}
&\to^* \{x_2 \doteq 0, y_2 \doteq y_1, z_2 \doteq y_1, x_3 \doteq s(x_2), y_2 \doteq y_3, z_3 \doteq s(z_2),\\
&\quad s(x_3) \doteq s^2(0), y_2 \doteq s^2(0), s(z_3) \doteq r\}\\
&\to^* \{y_2 \doteq y_1, z_2 \doteq y_1, x_3 \doteq s(0), y_2 \doteq y_3, z_3 \doteq s(z_2), s(x_3) \doteq s^2(0), y_2 \doteq s^2(0), s(z_3) \doteq r\}\\
&\quad \to^* \{z_2 \doteq y_1, x_3 \doteq s(0), y_1 \doteq y_3, z_3 \doteq s(z_2), s(x_3) \doteq s^2(0), y_2 \doteq s^2(0), s(z_3) \doteq r\}\\
&\quad\quad \to^* \{x_3 \doteq s(0), y_1 \doteq y_3, z_3 \doteq s(y_1), s(x_3) \doteq s^2(0), y_1 \doteq s^2(0), s(z_3) \doteq r\}\\
&\quad\quad\quad \to^* \{y_1 \doteq y_3, z_3 \doteq s(y_1), s(s(0)) \doteq s^2(0), y_1 \doteq s^2(0), s(z_3) \doteq r\}\\
&\quad\quad\quad\quad \to^* \{z_3 \doteq s(y_3), s(s(0)) \doteq s^2(0), y_3 \doteq s^2(0), s(z_3) \doteq r\}\\
&\quad\quad\quad\quad\quad \to^* \{s(s(0)) \doteq s^2(0), y_3 \doteq s^2(0), s(s(y_3)) \doteq r\}\\
&\quad\quad\quad\quad\quad\quad \to^* \{y_3 \doteq s^2(0), s(s(y_3)) \doteq r\}\\
&\quad\quad\quad\quad\quad\quad\quad \to^* \{s(s(s^2(0))) \doteq r\}\\
&\quad\quad\quad\quad\quad\quad\quad \to^* \{r \doteq s(s(s^2(0)))\}
\end{aligned}$$

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. This can be thought as a chemical reaction having an effect on the non-involved entities.*

The *normalisation* or *execution* $\texttt{Ex}(\Sigma)$ (figure 1) of a constellation Σ constructs the set of all possible correct (the underlying unification problem doesn't fail) saturated (no stars can be added to extend it) diagrams and actualise 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 [**?**] corresponding to a certain class of clauses we can infer using the resolution rule. 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 6. *For $\Sigma^{2,2}_{\mathbf{N}}$ (example 1), one can check that we have $\texttt{Ex}(\Sigma^{2,2}_{\mathbf{N}}) = [s^4(0)]$ because only the complete computation of example 3 succeed and all other saturated diagrams represents partial or over computations and fail.*

$$\Sigma \overset{\text{set of diagrams}}{\longrightarrow} \cup_{k=0}^{\infty} D_k \overset{\text{restriction}}{\longrightarrow} D'_1, ..., D'_n \subseteq \cup_{k=0}^{\infty} D_k \overset{\text{actualisation}}{\longrightarrow} \sigma_1 + ... + \sigma_n$$

Figure 1: Illustration of the execution of a strongly normalising constellation where the restriction only keep the correct and saturated diagrams. The resulting constellation is $\texttt{Ex}(\Sigma)$.

2 Interpretation of the computional content of MLL

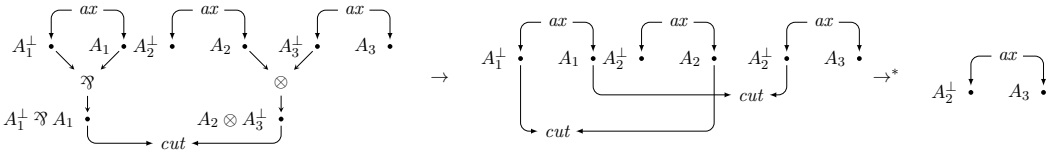
The idea comes from the Geometry of Interaction: the $\parr/\otimes$ cuts can be seen as administrative/inessential cuts since all they do is basically a rewiring on the premises of the $\parr$ and $\otimes$ nodes connected together. By eliminating all such cuts, we end up with a set of axioms connected by cuts. These axioms represent the essential and computational part of a proof-structure. To simulate the dynamic of cut-elimination, we consider proof-structures post multiplicative cut-elimination. We translate the axioms and the cuts into stars:

1. A formula A conclusion of axiom becomes a ray $+c.p_A(t)$ where t represents the "address" of A relatively to the conclusions of the proof-structure (without considering cuts).

2. An axiom becomes a binary star containing the address of its formulas. It is coloured with $c+$ for "positive computation".

3. A cut between A and $A^\perp$ becomes a binary star $[-c.p_A(x), -c.p_B(x)]$ it is coloured with $-c$ for "negative computation" in order to connect them to axioms.

Example 7. *Let's encode the following cut-elimination $\mathcal{S} \to^* \mathcal{S}'$ of MLL proof-structures:*

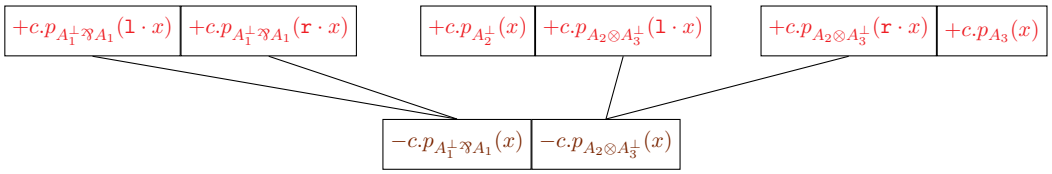


The address of $A_1^\perp$ is $p_{A_1^\perp \parr A_1}(\mathtt{l}\cdot x)$ because it is located on the left-hand side of $A_1^\perp \parr A_1$. The address of $A_3^\perp$ is $p_{A_2 \otimes A_3^\perp}(\mathtt{r}\cdot x)$ and the one for A_3 is $p_{A_3}(x)$. The proof-structure $\mathcal{S}$ is encoded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

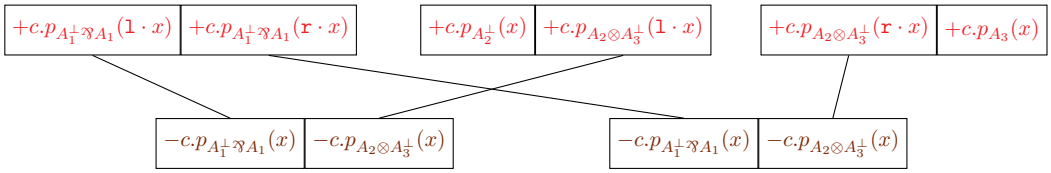
$$[+c.p_{A_1^\perp \parr A_1}(\mathtt{l}\cdot x), +c.p_{A_1^\perp \parr A_1}(\mathtt{r}\cdot x)] + [+c.p_{A_2^\perp}(x), +c.p_{A_2 \otimes A_3^\perp}(\mathtt{l}\cdot x)] +$$

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

and its unification graph is:



Since a ray can only be connected to a unique other ray, one occurrence of cut isn't sufficient in order to make a saturated diagram. We have to duplicate the cut star, which corresponds exactly to the second step of $\mathcal{S} \to^* \mathcal{S}'$. One can check that the only correct diagram is the following one (the left matches with the left and the right with the right):

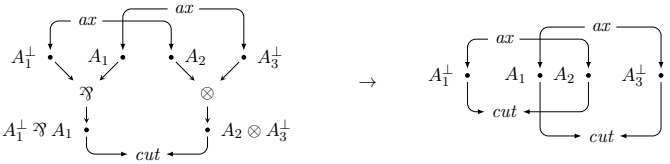


The matching is perfect so the connected rays are removed and we end up with

$$[+c.p_{A_2^\perp}(x), +c.p_{A_3}(x)]$$

corresponding to $\mathcal{S}'$. We can remark that in this case, the normalisation computes the set of all maximal paths. This corresponds to the interpretation of the cut-elimination in GoI.

Example 8. *If we have the following reduction $\mathcal{S} \to \mathcal{S}'$:*

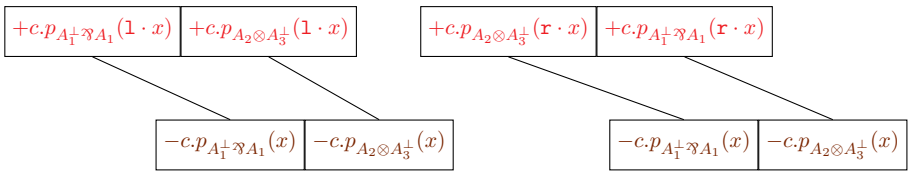


The constellation corresponding to $\mathcal{S}$ is

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

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

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



which contains two cycles and is the translation of $\mathcal{S}'$. These cycles can be unfolded as many times as we want by reusing some stars. It is impossible to leave a free ray since all diagrams can always be extended by adding further occurrences of stars. Therefore, all diagrams are closed and doesn't define a correct diagram (the star of free is undefined because no empty star can exist). We finally have $\mathtt{Ex}(\mathcal{S}) = \emptyset$.

3 Interpretation of the logical content of MLL

3.1 The correctness criterion of Danos-Regnier

We translate the correctness criterion of Danos-Regnier [**?**] into the stellar resolution to show that it can be described very naturally by the unification of terms.

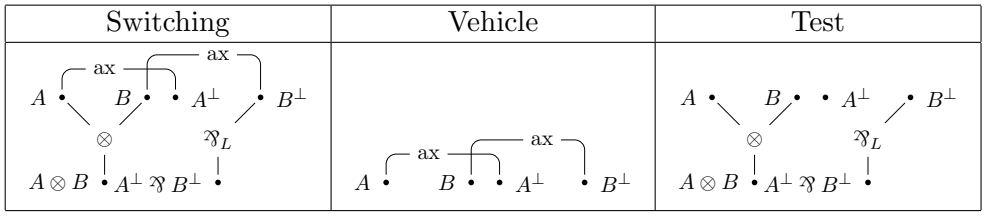
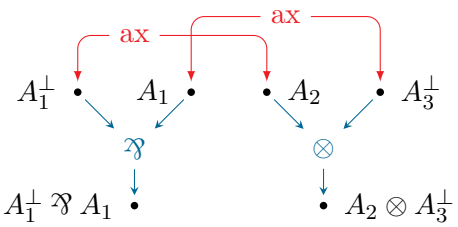


Figure 2: The vehicule and test corresponding to a Danos-Regnier switching graph.



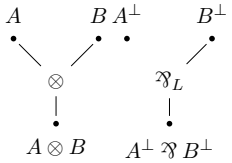
The upper part made of axioms appearing in a proof-structure is called the *vehicle* and the lower-part is the *gabarit*. With the Danos-Regnier switchings (called *ordeals*), only the lower-part (basically a syntax tree) is changed so a proof-structure may be divided into two parts: the vehicle is the *tested* and gabarit is a *set of test* against the vehicle. The vehicle holds the computational part of a proof and the test its logical part (type/formula). This test vehicule/gabarit produces a certification: if all tests pass, we have a *proof-net*. This is Girard's "usine". Note that testing is symmetric: a gabarit is a also tested by a vehicle.

Ordeals are translated in a very natural way by translating their nodes into constellations (the fractional notation is purely esthetical):

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

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 $[p_{A_1}(x), ..., p_{A_n}(x)]$.

Example 9. *Here is an example with the switching graph and the test of figure 2:*

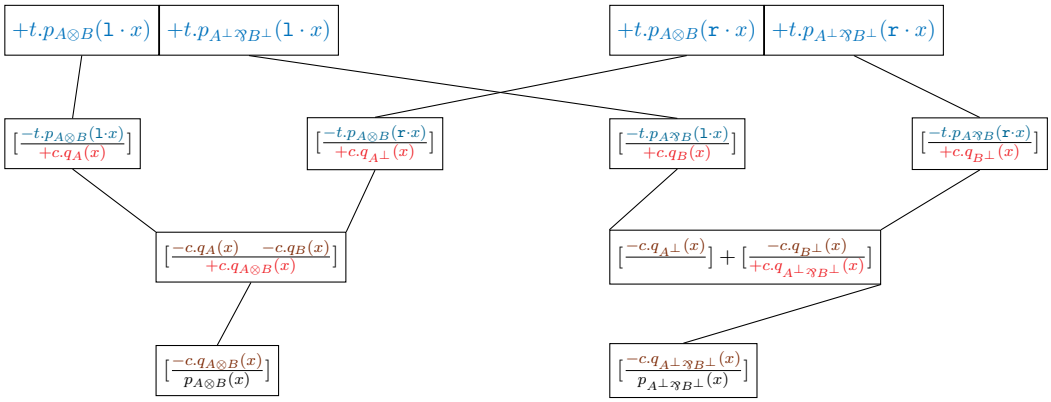


$$[\tfrac{-t.p_{A\otimes B}(\mathtt{l}\cdot x)}{+c.q_A(x)}] + [\tfrac{-t.p_{A\parr B}(\mathtt{l}\cdot x)}{+c.q_B(x)}] + [\tfrac{-t.p_{A\otimes B}(\mathtt{r}\cdot x)}{+c.q_{A^\perp}(x)}] + [\tfrac{-t.p_{A\parr B}(\mathtt{r}\cdot x)}{+c.q_{B^\perp}(x)}] +$$
$$[\tfrac{-c.q_A(x) \quad -c.q_B(x)}{+c.q_{A\otimes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

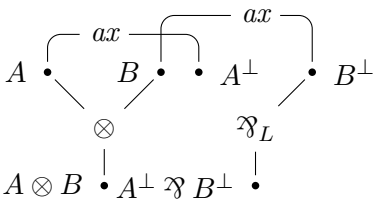
When connected to the vehicle

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

we obtain the following unification graph:



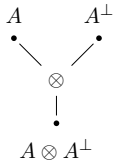
structurally corresponding to the following switching graph:



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

The essential point of the translation is that the corresponding unification graph will have the same shape as a switching graph.

Example 10. *If we have the following test instead:*



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

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 unification graph and since the matching is perfect, we can construct infinitely many correct diagrams. The constellation isn't strongly normalisable.

3.2 Interpretation of formulas

In order to reconstruct the types/formulas, we follow the construction of model of realisability. In the traditional type theory, types have a normative/constrictive role: they prevent some unwanted connexions to happen. In our reconstruction of types, similarly to realisabiliy, types have a descriptive role: they describe the common behaviour of a collection of computational objects (constellations in our case). We obtain a more refined idea of type but also a more complicated one to reason with. We can think of it as a kind of liberalised typing free from constraints but which is also very chaotic.

We work with set of constellations corresponding to kind of "pre-types".

Orthogonal. First, we need to choose a *criterion of orthogonality* opposing constellations. We choose the strong normalisation but others choices (leading to different idea of correctness) can be chosen. If we have a set of constellation $\mathbf{A}$, its orthogonal, written $\mathbf{A}^\perp$, is the set of all constellations which are strongly normalising when interacting with the constellations of $\mathbf{A}$.

Conducts. A *conduct* (or type/formula) $\mathbf{A}$ is a set of constellations orthogonal to another set $\mathbf{B}$ i.e $\mathbf{A} = \mathbf{B}^\perp$. It means that it interacts well (with respects to the orthogonality) with another pre-type. It is equivalent to say that $\mathbf{A} = \mathbf{A}^{\perp\perp}$. The intuition is that the interaction between a pre-type and its dual (seen as the set of its tests) is closed.

Atoms. We define atoms with a *basis of interpretation* Φ associating of each type variable X_i a distinct conduct $\Phi(X_i)$. It represents a choice of formula for each variable.

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. The conduct $\mathbf{A}$ and $\mathbf{B}$ have to be disjoint in the sense that they can't be connected together by two matching rays.

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

Example 11. *Let* $\mathbf{A} = \{[+a.x]\}$ *be a set containing one constellation.*

- $\mathtt{Ex}([+a.x] + [+b.x]) = \emptyset$ *therefore* $[+a.x] \perp [+b.x]$ *and* $[+b.x] \in \mathbf{A}^\perp$.
- $\mathtt{Ex}([+a.x] + [a.x, x]) = [x]$ *therefore* $[+a.x] \perp [-a.x, x]$ *and* $[-a.x, x] \in \mathbf{A}^\perp$.
- $\mathtt{Ex}([+a.x] + [-a.x, +a.x])$ *isn't strongly normalising therefore* $[-a.x, +a.x] \notin \mathbf{A}^\perp$.

Example 12. *When taking the translation $\Sigma \in \mathbf{A}$ of the vehicle of a proof-net of conclusion A, it is strongly normalising when interacting with the gabarit $\mathcal{G}$ corresponding to the set of constellations containing the ordeals for A. The set of all constellation with which it strongly normalises is $\mathbf{A}^\perp$. Therefore, we have $\mathcal{G} \subseteq \mathbf{A}^\perp$. It is a materialisation of the fact that the ordeals for $\mathbf{A}$ represent a partial proof of $\mathbf{A}^\perp$.*

Example 13. *Let $\Sigma^+_{\mathbf{N}} = [+add(0,y,y)] + [+add(s(x),y,s(z)), -add(x,y,z)]$ be a constellation. Let $\mathbf{QAdd} = \{[-add(s^n(0),s^m(0),r),r] \mid n,m \in \mathbf{N}\}$ and $\mathbf{AAdd} = \{[s^n(0)] \mid n \in \mathbf{N}\}$. Let's take a constellation $[-add(s^n(0),s^m(0),r),r]$ and connect it with $\Sigma^+_{\mathbf{N}}$. All diagrams corresponding to $\Sigma^+_{\mathbf{N}}$ 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 $\Sigma \in \mathbf{QAdd}$, $\mathtt{Ex}(\Sigma \cdot \Sigma^+_{\mathbf{N}}) \in \mathbf{AAdd}$ and $\Sigma^+_{\mathbf{N}} \perp \Sigma$. One can even check that we also have $\Sigma^+_{\mathbf{N}} \in \mathbf{QAdd}^{\perp\perp} \multimap \mathbf{AAdd}$. This provides a kind of naive typing for logic programs.*

References

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

[2] 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.

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

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

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

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

[7] Jean-Yves Girard. Geometry of interaction VI: a blueprint for transcendental syntax. *preprint*, 2013.

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

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

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

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

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

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

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

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

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

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