Geometry of Interaction, part 2: Unification and exponentials

Marc Bagnol

September 6, 2013

In the previous episode,

we saw a model of linear logic and its cut-elimination procedure based on "flows": links between two addresses. However, the interpretation was limited to the multiplicative part of logic: MLL.

In this second part of the talk, we will start by extending the notion of flow, which will allow us to extend the model to exponential connectives.

Contents

1 Limits of addresses 1

2 Unification 1

3 Flows 3

4 Interpretation of exponential connectives 4

5 Further reading 7

A MELL with functorial promotion 8

B Exercises 8

1 Limits of addresses

In the first part of the talk, the interpretation of proofs was based on partial injections over a finite set of addresses.

This was already unhandy when interpreting cut: we needed addresses with a "$\star$" as a shorthand, $\mathtt{l.r.}\star$ meaning "any address that starts with $\mathtt{l.r}$", this notion did not have a very clear status.

Moreover, we could only interpret η-expanded proofs.

But more importantly, we were unable to interpret exponential connectives, that deal with the infinitary part of logic. Through contraction in particular, they can involve any number of reuse of the same formula. To interpret them, a space that is potentially infinite is needed.

Still, we want something that remain concrete, in the sense of a finite syntax.

The unification technique provides a solution to these constraints. In a sense, it gives an official status to the "$\star$" shorthand, extanding greatly the expressivity of the language.

2 Unification

The unification was first introduced by Herbrand, then studied further in the 60's by Robinson, as part of his automated deduction procedure. It is also at the core of the PROLOG programming language.

Very generally, a unification problem consists in wondering if two terms in a certain language can be "made equal" by substituting their variables.

Terms and substitutions

Everything that follows is valid for any set of first-order terms. However, to make things a little more concrete we fix once and for all the set of terms we consider, given by the following grammar:

$$\mathbb{T} ::= x, y, z, \ldots \mid \mathtt{a}, \mathtt{b}, \mathtt{c}, \ldots \mid \mathbb{T}.\mathbb{T}$$

where $x, y, z, \ldots$ are the **variables** and $\mathtt{a}, \mathtt{b}, \mathtt{c}, \ldots$ are the **constants**. The set $\mathbb{V}$ of variables and the set $\mathbb{C}$ of constants are countably infinite.

As for the binary symbol "." we will write parenthesis as right associating: $x.y.\mathtt{d} := x.(y.\mathtt{d})$

A **substitution** is a mapping $\theta : \mathbb{V} \to \mathbb{T}$ from variables to terms, with finite **domain**, *i.e.* such that the set $\mathtt{Dom}(\theta) := \{ v \in \mathbb{V} \mid \theta(v) \neq v \}$ is finite.

Notation: *a substitution with domain* $\{ x_1, \ldots, x_n \}$ *such that* $\theta(x_1) = u_1, \ldots, \theta(x_n) = u_n$ *will be written as* $\{ x_1 \mapsto u_1 ; \ldots ; x_n \mapsto u_n \}$.

Substitutions act on terms in the expected way: $t.\{ x_i \mapsto u_i \} := t[u_i/x_i]$

One can define a **composition** operation on them that satisfies $t.(\theta;\theta') = (t.\theta).\theta'$: if $\theta = \{x_i \mapsto u_i\}$ and $\theta' = \{ y_j \mapsto v_j \}$, we set

$$\theta;\theta' := \{ x_i \mapsto u_i.\theta' \} \cup \{ y_i \mapsto v_i \mid y_i \notin \mathtt{Dom}(\theta) \}$$

Example:

$\theta = \{z \mapsto z.x \; ; \; x \mapsto \mathtt{c}\}$

$\theta;\theta = \{z \mapsto (z.x).\mathtt{c} \; ; \; x \mapsto \mathtt{c}\}$

$\theta;\theta;\theta = \{z \mapsto ((z.x).\mathtt{c}).\mathtt{c} \; ; \; x \mapsto \mathtt{c}\}$

A bit of vocabulary:

- A **renaming** is a substitution θ such that $\theta(\mathbb{V}) \subseteq \mathbb{V}$ and that is bijective.
- Two substitutions θ, θ' are equal **up to renaming** if there exist a renaming α such that $\theta' = \theta;\alpha$.
- A substitution θ' is an **instance** of θ if there exists a substitution σ such that $\theta' = \theta;\sigma$.

  It is easy to see that if two substitutions θ, θ' are instances of one another, then they are equal up to renaming.

Unification problems

Two terms t, u are **unifiable** if there exists a substitution θ such that

$$t.\theta = u.\theta$$

In that case, we write $t \sim u$ and we say that θ is a **unifier** of t and u.

We say moreover that a unifier θ of t, u is **principal** (also called a **most general unifier**, MGU) if any other unifier of t, u is an instance of θ. Therefore, principal unifiers of two terms are equal up to renaming.

Examples:

$\mathtt{f}.x \sim \mathtt{f}.\mathtt{c}$ *with* $\theta = \{x \mapsto \mathtt{c}\}$

$\mathtt{g}.\mathtt{f}.x \not\sim \mathtt{f}.\mathtt{f}.x$

$x.v \sim u.y$ *with* $\theta = \{x \mapsto u\,;\, y \mapsto v\}$

$x \not\sim \mathtt{f}.x$

$\mathtt{f}.x \sim \mathtt{f}.\mathtt{g}.y$ *with*

$\begin{cases} \theta = \{x \mapsto \mathtt{g}.\mathtt{g}.\mathtt{c}\,;\, y \mapsto \mathtt{g}.\mathtt{c}\} & \textit{(not principal)} \\ \textit{or } \theta' = \{x \mapsto \mathtt{g}.z\,;\, y \mapsto z\} & \textit{(principal)} \end{cases}$

An important feature of the theory of first-order unification is the (decidable) existence of principal unifiers in the case of unifiable terms.

Theorem 2.1 - MGU

If two terms are unifiable, then they have principal unifiers.
Wether two terms are unifiable and, in case they are, finding a MGU is a decidable problem.

3 Flows

With this new unification toolbox, we can now extend the notion of flow. Remember the previous version that was based on links from an address to another. Here we will follow the same idea, but instead of addresses we will use terms. Composition needs then to be refined to take unification into account.

Definition 3.1 - flow

A **flow** is a couple, written $t \leftarrow u$, of terms with the same set of variables.

Flows are considered *up to renaming*:

$$x \leftarrow x \;=\; y \leftarrow y \;=\; z \leftarrow z \;=\; \ldots$$
$$x.y \leftarrow y.x \;=\; z.x \leftarrow x.z \;=\; \ldots \;\neq\; x.y \leftarrow x.y$$

are examples of flows.

Before giving the definition of the (partial) composition of flows, let us point a usefull intuition about them: as they are considered up to renaming, the flows will indeed work by *matching* rather than unification. A flow $u \leftarrow v$ can in fact be thought of as a `'match ... with v -> u'` in a ML-style language. The composition of flows follows this pattern.

Definition 3.2 - composition

Let $l_1 = u \leftarrow v$ and $l_2 = t \leftarrow w$. Suppose we have chosen two representatives of the renaming classes such that their sets of variables are disjoint.

The **composition** (or **product**) of l_1 and l_2 is defined if $v \sim t$ with MGU θ and in that case

$$l_1\, l_2 := u.\theta \leftarrow w.\theta$$

Examples:

$(\mathtt{g}.x \leftarrow \mathtt{f}.\mathtt{f}.x)(\mathtt{f}.y \leftarrow \mathtt{f}.\mathtt{g}.y) = \mathtt{g}.x \leftarrow \mathtt{f}.\mathtt{g}.\mathtt{f}.x$

$\big(\mathtt{g}.x \leftarrow (\mathtt{f}.\mathtt{g}.x).(\mathtt{g}.\mathtt{c})\big)\,\big((\mathtt{f}.y).y \leftarrow \mathtt{f}.\mathtt{f}.y\big) = \mathtt{g}.\mathtt{c} \leftarrow \mathtt{f}.\mathtt{f}.\mathtt{g}.\mathtt{c}$

This operation respects the renaming classes because of the equivalence of MGUs up to renaming.

If we add a special flow $\perp$ for the cases of failed unification, we get a product of flows that is associative, and has a neutral element, $1 := x \leftarrow x$.

Moreover, we can define an involution $(.)^\dagger$ as $(u \leftarrow v)^\dagger := v \leftarrow u$ such that $ll^\dagger l = l$ and $l^\dagger l l^\dagger = l^\dagger$, making the set of flows an *inverse monoid*.

The unification semiring $\mathfrak{U}$

Last thing before we go on with the interpretation of linear logic: we need to consider not only flows, but rather (formal) sums of flows. We therefore extend our structure into a semiring: we consider sets of (non-$\bot$, which is replaced by the empty set) flows with a product defined as

$$U\,V := \{\, uv \mid u \in U,\, v \in V,\, uv \neq \bot \,\}$$

To insist on the algebraic structure, we will use $+$ instead of $\cup$ and 0 instead of $\varnothing$: so that $\{u_1,\dots,u_n\} = u_1 + \cdots + u_n$, $V + W = V \cup W$ … for instance $u + v + 0 = \{u\} \cup \{v\} \cup \varnothing = \{u, v\}$

Remark. *If we had wanted our structure to be in a more* quantitative *spirit, we could have chosen to consider* multisets *instead of sets of flows. But, as in the first part of the talk, in the cases we will consider this does not matter.*

Notation: *we will write* $u \rightleftharpoons v$ *for* $u \leftharpoonup v + v \leftharpoonup u$ *and* $\downarrow u$ *for* $u \leftharpoonup u$.

4 Interpretation of exponential connectives

The general pattern remains the same: the interpretation of a proof[1] of conclusion $\vdash A_1,\dots,A_n$ with cuts on the pairs of formulae $\{(C_1, C_1^{\perp}),\dots,(C_n, C_n^{\perp})\}$ is a triple

$$\left(\{\mathtt{A}_1,\dots,\mathtt{A}_n\}\ ,\ \{(\mathtt{C}_1,\mathtt{C}_1^{\perp}),\dots,(\mathtt{C}_n,\mathtt{C}_n^{\perp})\}\ ,\ V \right)$$

the $\mathtt{A}_i, \mathtt{C}_i, \mathtt{C}_i^{\perp}$ being constant symbols and V being a sum of flows of the form '$\mathtt{X}.u \leftharpoonup Y.v$' ($X, Y$ being either $\mathtt{A}_i$, $\mathtt{C}$ or $\mathtt{C}^{\perp}$) with u, v being now terms in our new language, no longer simply addresses.

As for the multiplicative part, the interpretation does not change much. The only potential modification is at the level of axioms to allow the interpretation of non η-expanded proofs. This is left as an exercise (see the appendix).

On the other hand, the interpretation of exponential rules will use fully our new language.

The general idea is that to any promotion rule (*i.e.* any box in the language of proofnets) will correspond a distinct variable y. Each of these variables allow to have several flows that

The exponential depth will be handled by using "." as an actual binary function and no longer simply as a way to write sequences of letters.

Let us now sketch the interpretation, in the cut-free case for better readability.

Promotion

This is where new variables appear: if we have π of conclusion $\vdash A_1,...,A_n,B$ and π' of conclusion $\vdash ?A_1,...,?A_n,!B$ obtained from π by a promotion rule, then $[\pi']$ is obtained from $[\pi]$ by selecting a fresh variable y and

- replacing any term of the form $\mathtt{A}_i.u$ by $?\mathtt{A}_i.u.y$
- replacing any term of the form $\mathtt{B}.u$ by $!\mathtt{B}.u.y$

Digging[2]

If we have π of conclusion $\vdash \Gamma, ??A$ and π' of conclusion $\vdash \Gamma, ?A$ obtained from π by a digging rule, then $[\pi']$ is obtained from $[\pi]$ by

- replacing any term of the form $??\mathtt{A}.(u.v).w$ by $?\mathtt{A}.u.(v.w)$

[1] We can either consider sequent proofs or proofnets. Indeed, two sequent proofs equated when translated as proofnets will be equated when translated as cut systems, so that it does not really matter.

[2] In these cases, we have by induction that the terms have the proper form.

Contraction[2]

If we have π of conclusion $\vdash \Gamma, ?A_1, ?A_2$ and π' of conclusion $\vdash \Gamma, ?A$ obtained from π by a contraction rule, then $[\pi']$ is obtained from $[\pi]$ by

- replacing any term of the form $?\mathtt{A}_1.u.v$ by $?\mathtt{A}.u.(\mathtt{l}.v)$
- replacing any term of the form $?\mathtt{A}_2.u.v$ by $?\mathtt{A}.u.(\mathtt{r}.v)$

Dereliction

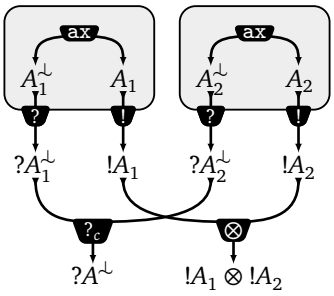
If we have π of conclusion $\vdash \Gamma, A$ and π' of conclusion $\vdash \Gamma, ?A$ obtained from π by a dereliction rule, then $[\pi']$ is obtained from $[\pi]$ by

- replacing any term of the form $\mathtt{A}.u$ by $?\mathtt{A}.u.\mathtt{d}$

Weakening

There is nothing to do here: if we have π of conclusion $\vdash \Gamma$ and π' of conclusion $\vdash \Gamma, ?A$ obtained from π by a weakening rule, then $[\pi']$ is $[\pi]$, with a new constant symbol $?\mathtt{A}$.

Example: *the following proofnet*



is interpreted as

$$?\mathtt{A}^\perp.x.l.y \rightleftharpoons (!\mathtt{A}_1 \otimes !A_2).l.x.y \;+\; ?\mathtt{A}^\perp.x.r.y \rightleftharpoons (!\mathtt{A}_1 \otimes !\mathtt{A}_2).r.x.y$$

Execution of cuts

Here again the idea does not change much: the result of executing a subset of the cuts is given by the **execution formula** which contains a possibly infinite sum.

Definition 4.1 - flows and cuts

Let $(\Gamma, \mathfrak{C}, V)$ be a cut system and $\mathfrak{D} \subseteq \mathfrak{C}$. We associate to it the following two sum of flows:

$$\sigma_{\mathfrak{D}} := \sum_{\mathtt{D},\mathtt{D}^\perp \in \mathfrak{D}} \mathtt{D}_i.x \rightleftharpoons \mathtt{D}_i^\perp.x \qquad (1 - p_{\mathfrak{D}}) := \sum_{\mathtt{A} \in \Gamma} \downarrow \mathtt{A}.x \;+ \sum_{\mathtt{C},\mathtt{C}^\perp \in \mathfrak{C} \setminus \mathfrak{D}} \downarrow \mathtt{C}.x + \downarrow \mathtt{C}^\perp.x$$

Definition 4.2 - execution formula

Let $S = (\Gamma, \mathfrak{C}, V)$ a cut system and $\mathfrak{D} \subseteq \mathfrak{C}$.

The **execution** of S with respect to $\mathfrak{D}$ is $[\mathfrak{D}]S := (\Gamma, \mathfrak{C} \setminus \mathfrak{D}, [\mathfrak{D}]V)$ with $[\mathfrak{D}]V$ defined, when $\sigma_{\mathfrak{D}} V$ is nilpotent, as

$$(1 - p_{\mathfrak{D}}) \left(V + V(\sigma_{\mathfrak{D}} V) + V(\sigma_{\mathfrak{D}} V)^2 + V(\sigma_{\mathfrak{D}} V)^3 + \cdots \right) (1 - p_{\mathfrak{D}})$$

The intuition is the same that in the multiplicative case: we are algebraically listing all paths on a graph: we go through alternately the axioms (V) and the cuts that we are executing ($\sigma_{\mathfrak{D}}$), until there is no more path to explore (nilpotency). Then we restrict the result to the complement of the cut symbols (multiplication by $(1 - p_{\mathfrak{D}})$).

It still enjoys the associativity/Church-Rosser property.

Theorem 4.3 - associativity [1]

Let $S = (\Gamma, \mathfrak{C} \cup \mathfrak{D} \cup \mathfrak{E}, V)$ be a cut system. $[\mathfrak{C}]([\mathfrak{D}]S)$ is defined if and only if $[\mathfrak{C} \cup \mathfrak{D}]S$ is, and in that case

$$[\mathfrak{C}]([\mathfrak{D}]S) = [\mathfrak{C} \cup \mathfrak{D}]S$$

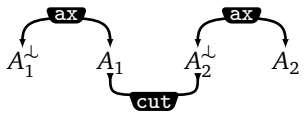
Moreover, the cut-systems that are the interpretation of proofs have always their execution defined:

Theorem 4.4 - nilpotency [1]

If $(\Gamma, \mathfrak{C}, V)$ is the interpretation of a MELL proof and $\mathfrak{D} \subseteq \mathfrak{C}$, then $\sigma_{\mathfrak{D}} V$ is nilpotent.

Remark. *This theorem is, in the language of GoI, a result of* strong normalisation. *It is proven by techniques based on orthogonality and reducibility.*

Let us see on an example how it works: consider the cut system associated to the proofnet



We have $V = \mathtt{A}_1.x \rightleftharpoons \mathtt{A}_1^{\perp}.x + \mathtt{A}_2.x \rightleftharpoons \mathtt{A}_2^{\perp}.x$, $\sigma_{\mathfrak{C}} = \mathtt{A}_1.x \rightleftharpoons \mathtt{A}_2^{\perp}.x$ and $p_{\mathfrak{C}} = \downharpoonright \mathtt{A}_1^{\perp}.x + \downharpoonright \mathtt{A}_2.x$.

So that $\sigma_{\mathfrak{C}} V = \mathtt{A}_1.x^{\perp} \leftharpoonup \mathtt{A}_2^{\perp}.x + \mathtt{A}_2.x \leftharpoonup \mathtt{A}_1.x$ and with $(\sigma_{\mathfrak{C}} V)^2 = 0$. Therefore $[\mathfrak{C}]V = \mathtt{A}_1^{\perp}.x \rightleftharpoons \mathtt{A}_2$, which corresponds to the interpretation of the cut-free proofnet obtained from R.

However, if on this example the interpretation seems sound with respect to cut-elimination, not everything is perfect. The GoI approach to logic is based from the start on a *local* reduction procedure: execution with respect to a certain cut-pair $(\mathtt{C}, \mathtt{C}^{\perp})$ cannot affect flows with none of their terms that start with $\mathtt{C}$ or $\mathtt{C}^{\perp}$; while some exponential reduction steps require rather *global* operations.

The soundness theorem we have in the end is therefore limited. The problem is really with the auxiliary doors of promotion boxes, so a way to avoid the problem is to exclude formulae containing ? from the conclusions (it can still appear in cuts).

Theorem 4.5 - limited soundness [1]

> Let $[\pi] = (\Gamma, \mathfrak{C}, V)$ be the interpretation of a MELL proof. If Γ does not contain any "?" connective and π' is the cut-free proof obtained from π, we have
>
> $$[\pi'] = (\Gamma, \varnothing, [\mathfrak{C}]V)$$

An example of mismatch in the general case is treated in an exercise.

We don't have time to discuss this point in full details, but a remark can still be made: even if there is a mismatch, the calculus is still sound, in the sense that a complex function with a "?"-free output (for instance boolean) implemented in MELL or in GoI will give the same results (by associativity) even if the implementations of the function itself may differ.

5 Further reading

We cannot aim at exhaustivity on the subject in just one talk, so the interested reader should refer to:

- [1] for the interpretation of exponential connectives, in a different language, and the proofs of the main theorems: associativity, nilpotency and limited soundness.
- [6] for more details about unification. Also there are some notes I wrote (in French!) on the unification algebra: http://iml.univ-mrs.fr/~bagnol/notes/unification.pdf.
- The algebraic point of view can be pushed further to the point of interpreting logic in Von Neumann algebras, these are rather difficult reads, but if you are curious, see [2, 3].
- On the more traditionnal side, it is possible to integrate the notions of switchings from proofnets theory into the GoI interpretation [4].
- Another point of view on the interpretation is that of token machines, see [5] (chapter 8).

References

[1] Jean Y. Girard. Geometry of interaction I: Interpretation of System F. In C. Bonotto, editor, *Logic Colloquium '88*, pages 221–260. North Holland, 1989.

[2] Jean-Yves Girard. Geometry of interaction IV : the feedback equation. In Väänänen Stoltenberg-Hansen, editor, *Logic Colloquium '03*, pages 76 – 117. Association for Symbolic Logic, 2006.

[3] Jean-Yves Girard. Geometry of interaction V: Logic in the hyperfinite factor. *Theor. Comput. Sci.*, 412(20):1860–1883, April 2011.

[4] Jean-Yves Girard. Geometry of interaction VI: a blueprint for transcendental syntax. Preprint.

[5] Olivier Laurent. *Étude de la polarisation en logique*. Thèse de doctorat, Université Aix-Marseille II, 2002.

[6] Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Trans. Program. Lang. Syst.*, 1982.

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

A MELL with functorial promotion

Identity group

$$\frac{}{\vdash A^{\perp}, A}\ (\text{ax}) \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, A^{\perp}}{\vdash \Gamma, \Delta}\ (\text{cut})$$

Multiplicatives

$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\ (\otimes) \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\ (\parr)$$

Exponentials

$$\frac{\vdash A_1, \dots, A_n, B}{\vdash ?A_1, \dots, ?A_n, !B}\ (!) \qquad \frac{\vdash \Gamma, ??A}{\vdash \Gamma, ?A}\ (??)$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\ (\text{d}) \qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\ (\text{c}) \qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?A}\ (\text{w})$$

B Exercises

Unification

Substitutions

Let $\alpha := \{x \mapsto y \ ; \ y \mapsto \mathtt{f.b} \ ; \ z \mapsto \mathtt{c}\}$ and $\beta := \{x \mapsto \mathtt{f.f}.x \ ; \ z \mapsto x \ ; \ y \mapsto \mathtt{g}.y\}$ be two substitutions.

- compute $(\alpha;\alpha)$, $(\alpha;\beta)$, $(\beta;\alpha)$, $(\beta;\beta)$

A substitution θ is said to be **idempotent** if $\theta;\theta = \theta$.

- Give a necessary and sufficient condition (in terms of domain) for a substitution to be idempotent
- Show that any substitution is equivalent up to renaming to an idempotent substitution

MGU

Solve (*i.e.* determine wether the two terms are unifiable and if they are, give a MGU) the following unification problems:

- $\mathtt{f}.(\mathtt{g}.x).y.(\mathtt{g}.x) \sim^? \mathtt{f}.y.(\mathtt{h}.z).(\mathtt{g}.w)$
- $\mathtt{f}.(\mathtt{g}.x).y.(\mathtt{g}.x) \sim^? \mathtt{f}.v.(\mathtt{h}.z).(\mathtt{g}.w)$
- $\mathtt{f}.x.(\mathtt{h}.x) \sim^? \mathtt{f}.(\mathtt{g}.y).z$
- $(\mathtt{h}.x.y).z \sim^? z.(\mathtt{h}.(\mathtt{f}.u).w)$
- $(\mathtt{h}.x.x).z \sim^? z.(\mathtt{h.f.g})$

Same question considering them as *matching* problems: the variables of the two terms can be instantiated differently, so that for instance x and $\mathtt{f}.x$ are matchable by $\{x \mapsto \mathtt{f}.x\}$ and $\{x \mapsto x\}$.

Instances

Prove that if two substitutions θ, θ' are instances of one another, then they are equal up to renaming.

Composition of flows

Compute the composition of the following flows:

- $(y.x \leftarrow x.y.\mathtt{g})((\mathtt{g}.\mathtt{h}).y.y \leftarrow y.\mathtt{h}) = ?$
- $(z.\mathtt{h} \leftarrow z.\mathtt{h}.z)(\mathtt{g}.x.\mathtt{f} \leftarrow x) = ?$
- $(x.x \leftarrow x.x)((\mathtt{f}.x).(\mathtt{f}.\mathtt{c}) \leftarrow x) = ?$

Axioms

Modify the interpretation of the axiom rule from the first talk to get an interpretation for non η-expanded axioms.

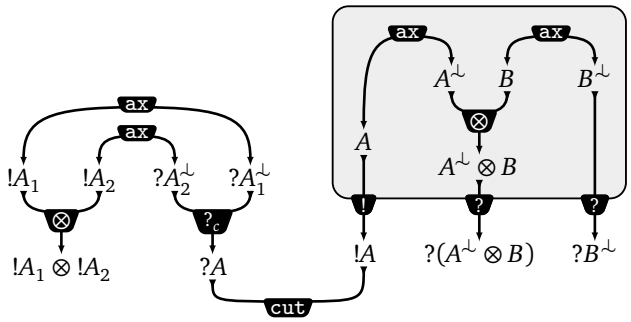
- First consider only MLL, and use unification to extend the interpretation.
- Then give a construction for MELL axioms. You will need to be careful with variables and nesting of exponential and multiplicative connectives.

Interpretation

Check the two examples of interpretation given in section 4 of the talk.

Mismatch

Give the interpretation of the following proofnet



and compute its GoI executions. Then compare it to the interpretation of the cut-free proofnet obtained by the usual cut-elimination procedure.

Towards $\Lambda^\star$

In the talk we took a very concrete approach to the GoI interpretation of MELL: we gave directly a model based on the unification semiring. However, it is possible to give a set of axioms that are sufficient for a mathematical structure to be a GoI model, this set of axiom is usually called $\Lambda^\star$ [1, 7].

Let us have a look at some properties that are satisfied by the unification semiring $\mathfrak{U}$ that are related to the axioms $\Lambda^\star$.

- define two elements L, R of $\mathfrak{U}$ such that $L^\dagger L = R^\dagger R = 1$ and $R^\dagger L = L^\dagger R = 0$
- define an injective **internal tensor product** over $\mathfrak{U}$, *i.e.* an injective binary function $(.) \otimes (.)$ that satisfies $(u+v) \otimes w = u \otimes w + v \otimes w$ and $w \otimes (u+v) = w \otimes u + w \otimes v$
- define an injective homomorphism $!(.)$ of $\mathfrak{U}$, together with two elements T, D of $\mathfrak{U}$ satisfying $T^\dagger(!!u)T = !u$ and $D^\dagger(!u)D = u$ for all u