Logical Constants from a Computational Point of View: Towards an Untyped Setting

Alberto Naibo[1] Mattia Petrolo[2]
Thomas Seiller[3]

[1]Université Paris 1 - EXeCO

[2]Université Paris 7 - Università Roma Tre

[3]Institut de Mathématiques de Luminy

January 21, 2011
Rencontre LOCI
Ludique et Sémantique de l'énoncé, CIRM

Outline

Outline

Inferentialism and Wittgenstein on linguistic use

- Semantics is not given by the denotation of a linguistic entity, but by its (correct) use in the language

The meaning of a word is its use in the language (Philosophical Investigations, §43)

- Cosequences fix the interpretation by making intentionality explicit

By "intention" I mean here what uses a sign in a thought. The intention seems to interpret, to give the final interpretation; which is not a further sign or picture, but something else ? the thing that cannot be further interpreted. But what we have reached is a psychological, not a logical terminus. (Philosophical Grammar, Part I, §98)

What are you telling me when you use the words . . .? What can I do with this utterance? What consequences does it have? (Last Writings I, §624)

Inferentialism and Wittgenstein on linguistic use

- Semantics is not given by the denotation of a linguistic entity, but by its (correct) use in the language

The meaning of a word is its use in the language (Philosophical Investigations, §43)

- Cosequences fix the interpretation by making intentionality explicit

By "intention" I mean here what uses a sign in a thought. The intention seems to interpret, to give the final interpretation; which is not a further sign or picture, but something else ? the thing that cannot be further interpreted. But what we have reached is a psychological, not a logical terminus. (Philosophical Grammar, Part I, §98)

What are you telling me when you use the words . . .? What can I do with this utterance? What consequences does it have? (Last Writings I, §624)

Dummett and Prawitz on linguistic use

Crudely expressed, there are always two aspects of the use of a given form of sentence: the conditions under which an utterance of that sentence is appropriate, which include, in the case of an assertoric sentence, what counts as an acceptable ground for asserting it; and the consequences of an utterance of it, which comprise both what the speaker commits himself to by the utterance and the appropriate response on the part of the hearer, including, in the case of assertion, what he is entitled to infer from it if he accepts it (Dummett 1973)

I shall [...] review some approaches to meaning that are based on how we use sentences in proofs. One advantage of such an approach is that from the beginning meaning is connected with aspects of linguistic use. (Prawitz 2006)

Dummett and Prawitz on linguistic use

Crudely expressed, there are always two aspects of the use of a given form of sentence: the conditions under which an utterance of that sentence is appropriate, which include, in the case of an assertoric sentence, what counts as an acceptable ground for asserting it; and the consequences of an utterance of it, which comprise both what the speaker commits himself to by the utterance and the appropriate response on the part of the hearer, including, in the case of assertion, what he is entitled to infer from it if he accepts it (Dummett 1973)

I shall [...] review some approaches to meaning that are based on how we use sentences in proofs. One advantage of such an approach is that from the beginning meaning is connected with aspects of linguistic use. (Prawitz 2006)

Logical inferentialism

Key ideas

- Semantics is not given by the denotation of a linguistic entity, but by its (correct) use in the language: in logic and formal systems this corresponds to assigning a semantic rôle to the deductive and proof-theoretic aspects.
- The meaning of logical constants is determined by the *inferential rules* that govern their use.

A problem (Prior 1960)

- `tonk` connective shows that some constraints are needed in order to define correctly the meaning of logical constants.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\text{Ax}}{A \vdash A\,\texttt{tonk}\,B}\,\texttt{tonk-intro}_1}{A \vdash B}\,\texttt{tonk-elim}_2}{\vdash A \Rightarrow B}\,{\Rightarrow}\text{-intro} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash B}\,\text{Ax}}{B \vdash A\,\texttt{tonk}\,B}\,\texttt{tonk-intro}_2}{B \vdash A}\,\texttt{tonk-elim}_1}{\vdash B \Rightarrow A}\,{\Rightarrow}\text{-intro}}{\dfrac{\vdash (A \Rightarrow B) \wedge (B \Rightarrow A)}{\vdash A \Leftrightarrow B}}\,{\wedge}\text{-intro}$$

Logical inferentialism

A solution (Dummett 1973)

- The conditions under which a given logical constant can be asserted should be in *harmony* with the consequences one can draw from the same logical constant.

  - Which set of rules has semantic priority ?

  1) Intro-rules (Gentzen/Prawitz/Tennant)
  2) Elim-rules (Martin-Löf[1970]/Schroeder-Heister[1985]/Dummett[1991])
  3) Either intro OR elim-rules (Milne/Rumfitt)
  4) The set of all rules (Brandom)

  - From a formal point of view, harmony has been presented in different ways

  1) conservativeness (Belnap/Kremer)
  2) normalization:
     2a) inversion principle (Prawitz)
     2b) general inversion principle (Negri/von Plato)
  3) Deductive equilibrium (Tennant)

Logical inferentialism

A solution (Dummett 1973)

- The conditions under which a given logical constant can be asserted should be in *harmony* with the consequences one can draw from the same logical constant.

  - Which set of rules has semantic priority ?

  1) Intro-rules (Gentzen/Prawitz/Tennant)
  2) Elim-rules (Martin-Löf[1970]/Schroeder-Heister[1985]/Dummett[1991])
  3) Either intro OR elim-rules (Milne/Rumfitt)
  4) The set of all rules (Brandom)

  - From a formal point of view, harmony has been presented in different ways

  1) conservativeness (Belnap/Kremer)
  2) normalization:
     2a) inversion principle (Prawitz)
     2b) general inversion principle (Negri/von Plato)
  3) Deductive equilibrium (Tennant)

Logical inferentialism

A solution (Dummett 1973)

- The conditions under which a given logical constant can be asserted should be in *harmony* with the consequences one can draw from the same logical constant.

  - Which set of rules has semantic priority ?

  1) Intro-rules (Gentzen/Prawitz/Tennant)
  2) Elim-rules (Martin-Löf[1970]/Schroeder-Heister[1985]/Dummett[1991])
  3) Either intro OR elim-rules (Milne/Rumfitt)
  4) The set of all rules (Brandom)

  - From a formal point of view, harmony has been presented in different ways

  1) conservativeness (Belnap/Kremer)
  2) normalization:
     2a) inversion principle (Prawitz)
     2b) general inversion principle (Negri/von Plato)
  3) Deductive equilibrium (Tennant)

Logical inferentialism

A solution (Dummett 1973)

- The conditions under which a given logical constant can be asserted should be in *harmony* with the consequences one can draw from the same logical constant.

- We focus on the formalization of harmony as *normalization*, that correspond to the so-called *Prawitz's inversion principle* (Prawitz 1973):

The elimination rules for a certain connective can never allow to deduce more than what follows from the direct grounds of its introduction rules.

- Such a condition bans tonk

$$\dfrac{\dfrac{\dfrac{\mathcal{D}}{\Gamma \vdash A}}{\Gamma \vdash A \,\texttt{tonk}\, B}\;\texttt{tonk-intro}}{\Gamma \vdash B}\;\texttt{tonk-elim} \qquad \rightsquigarrow \qquad ?$$

- It is impossible to define a normalization strategy.

Logical inferentialism

A solution (Dummett 1973)

- The conditions under which a given logical constant can be asserted should be in *harmony* with the consequences one can draw from the same logical constant.

- We focus on the formalization of harmony as *normalization*, that correspond to the so-called *Prawitz's inversion principle* (Prawitz 1973):

The elimination rules for a certain connective can never allow to deduce more than what follows from the direct grounds of its introduction rules.

- Such a condition bans tonk

$$\dfrac{\dfrac{\dfrac{\mathcal{D}}{\Gamma \vdash A}}{\Gamma \vdash A \,\texttt{tonk}\, B}\,\texttt{tonk-intro}}{\Gamma \vdash B}\,\texttt{tonk-elim} \qquad \rightsquigarrow \qquad ?$$

- It is impossible to define a normalization strategy.

A problem with *harmony-as-normalization*: modesty

- The condition of *harmony-as-normalization* does not ban all “tonkish" connectives: «harmony is an excessively modest demand» (Dummett 1991, p. 287).

- Let us add a new logical connective ($\ast$) to NJ through the following rules:

$$\ast\text{-intro}\ \frac{\Gamma \vdash A \qquad \Gamma' \vdash B}{\Gamma, \Gamma' \vdash A \ast B} \qquad\qquad \frac{\Gamma \vdash A \ast B \qquad \Gamma' \vdash A}{\Gamma, \Gamma' \vdash B}\ \ast\text{-elim}$$

- These rules enjoy a normalization strategy:

$$\ast\text{-elim}\ \dfrac{\ast\text{-intro}\ \dfrac{\dfrac{\mathcal{D}}{\Gamma \vdash A} \qquad \dfrac{\mathcal{D}_1}{\Gamma' \vdash B}}{\Gamma, \Gamma' \vdash A \ast B} \qquad \dfrac{\mathcal{D}_2}{\Gamma'' \vdash A}}{\Gamma, \Gamma', \Gamma'' \vdash B} \quad \rightsquigarrow \quad \dfrac{\mathcal{D}_1'}{\Gamma, \Gamma', \Gamma'' \vdash B}$$

Where $\mathcal{D}_1'$ is obtained by $\mathcal{D}_1$ by adjunction of Γ and Γ'' in the axioms.

A problem with *harmony-as-normalization*: modesty

- The ✶-connective does not enjoy the property of *deducibility of identicals* (Hacking 1979), i.e. it is not possible to prove $A \star B$ starting from the only assumption $A \star B$ with a non-trivial proof.

- Note that such a condition holds for other connectives, e.g.

$$\dfrac{\dfrac{\dfrac{}{A \Rightarrow B \vdash A \Rightarrow B}\,\text{Ax} \qquad \dfrac{}{A \vdash A}\,\text{Ax}}{A \Rightarrow B, A \vdash B}\,\Rightarrow\text{-elim}}{A \Rightarrow B \vdash A \Rightarrow B}\,\Rightarrow\text{-intro}$$

$$\dfrac{\dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}\,\text{Ax}}{A \wedge B \vdash A}\,\wedge\text{-elim}_1 \qquad \dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}\,\text{Ax}}{A \wedge B \vdash B}\,\wedge\text{-elim}_2}{A \wedge B \vdash A \wedge B}\,\wedge\text{-intro}$$

- This procedure fails for ✶:

$$\dfrac{\dfrac{\dfrac{}{A \star B \vdash A \star B}\,\text{Ax} \qquad \dfrac{}{A \vdash A}\,\text{Ax}}{A \star B, A \vdash B}\,\star\text{-elim}}{?}$$

A problem with *harmony-as-normalization*: modesty

- In the Sequent Calculus setting, this property of deducibility of identicals corresponds to the so-called atomic 'axiom-expansion' procedure. Again, for $\Rightarrow$ we have:

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\mathrm{Ax} \qquad \dfrac{}{B \vdash B}\,\mathrm{Ax}}{A \Rightarrow B, A \vdash B}\,\Rightarrow_L}{A \Rightarrow B \vdash A \Rightarrow B}\,\Rightarrow_R$$

- The absence of this property for $\ast$ indicates that the meaning of a connective is not only given by right and left rules but also by the axiom of the form $A \ast B \vdash A \ast B$.

- Indeed, the meaning of $\ast$ is not only given by its use (inferential rules) but also by some extra stipulation.

A problem with *harmony-as-normalization*: double-dealing

- Prawitz's inversion principle plays a double rôle:
  1. It is a meaning-condition: if (the definition of) a connective does not satisfy it, then it is not meaningful;
  2. It is a sufficient condition for being a logical constant: if a connective does not satisfy normalization, then it is not a logical constant.

- The risk is to identify two questions:
  1) What counts as the meaning of a linguistic connective;
  2) What counts as a logical constant.

- Not being a logical constant should not imply the fact of not being meaningful at all: it seems to be reasonable to have meaningful connectives that are not logical constants.

A possible solution

- We claim that the questions 1) and 2) belong to different domains of analysis.
- In particular, our proposal is that the analysis of what counts as a logical constant can be performed on a different level other the linguistic one, namely the *computational* one.
- We will show that the both the inversion principle and the deducibility of identicals can be interpreted as a computational properties.
- This guarantees the possibility of (partially) founding logical properties over computational ones: the lack of computational properties is sufficient for ruling out what is not logical.
- In order to develop our proposal we have to find a suitable setting for analyzing the notion of computation. A reasonable one seems to be λ-calculus.

Outline

Curry-Howard isomorphism

- λ-terms t are considered as programs; a type judgement $t : A$ is a program equipped with a specification that describes its behavior.
- The β-reduction corresponds to the execution of a program t, when applied to an argument u; the reduction shows how t computes.
- The Curry-Howard isomorphism establishes a one-to-one correspondance between Natural Deduction and λ-calculus, e.g.

$$\dfrac{\dfrac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \Rightarrow B}\,\Rightarrow\text{-intro} \qquad \Gamma' \vdash u : A}{\Gamma, \Gamma' \vdash (\lambda x.t)u : B}\,\Rightarrow\text{-elim} \qquad \rightsquigarrow \qquad \Gamma, \Gamma' \vdash t[{}^{u}/_{x}] : B$$

$$\dfrac{\dfrac{\Gamma \vdash t : A \qquad \Gamma' \vdash u : B}{\Gamma, \Gamma' \vdash \langle t, u \rangle : A \wedge B}\,\wedge\text{-intro}}{\Gamma, \Gamma' \vdash \pi_1(\langle t, u \rangle) : A}\,\wedge\text{-elim} \qquad \rightsquigarrow \qquad \Gamma \vdash t : A$$

Indeed, *normalization* in NJ corresponds to β-reduction in λ-calculus.

η-expansion

- In λ-calculus the main objects are programs, which are *intensional objects*: even if two programs compute the same mathematical functions, usually they are not considered as identical (e.g. one can be more efficient than the other).

- This means that there exist two terms t and t', $(t)u \equiv_\beta (t')u$ for all terms u, but not $t \equiv_\beta t'$.

- In order to work in the usual extensional setting, the following rules (η-*expansion*) are needed:

$$t \longrightarrow_\eta \lambda x(t)x$$

(with $x \notin FV(t)$)

$$t \longrightarrow_\eta \langle \pi_1(t), \pi_2(t) \rangle$$

- The relation of η-expansion is type-preserving.

η-expansion and deducibility of identicals

- η-expansion corresponds exactly to the property of deducibility of identicals:

$$\dfrac{\dfrac{\dfrac{}{t : A \Rightarrow B \vdash t : A \Rightarrow B}\,\text{Ax} \qquad \dfrac{}{x : A \vdash x : A}\,\text{Ax}}{t : A \Rightarrow B, x : A \vdash (t)x : B}\,\Rightarrow\text{-elim}}{t : A \Rightarrow B \vdash \lambda x(t)x : A \Rightarrow B}\,\Rightarrow\text{-intro}$$

$$\dfrac{\dfrac{\dfrac{}{t : A \wedge B \vdash t : A \wedge B}\,\text{Ax}}{t : A \wedge B \vdash \pi_1(t) : A}\,\wedge\text{-elim}_1 \qquad \dfrac{\dfrac{}{t : A \wedge B \vdash t : A \wedge B}\,\text{Ax}}{t : A \wedge B \vdash \pi_2(t) : B}\,\wedge\text{-elim}_2}{t : A \wedge B \vdash \langle \pi_1(t), \pi_2(t) \rangle : A \wedge B}\,\wedge\text{-intro}$$

Extensionality in λ-calculus

- We can define $\beta\eta$-equivalence ($\equiv_{\beta\eta}$) as the smallest equivalence relation containing $\longrightarrow_\beta$ and $\longrightarrow_\eta$.
- **Extensionality**: If t and t' are such that $(t)u \equiv_{\beta\eta} (t')u$ for all terms u, then $t \equiv_{\beta\eta} t'$
- Can we add some other type-preserving relation on λ-terms?

Maximality of $\equiv_{\beta\eta}$

- The answer is no. It is a consequence of **Böhm's Theorem**.

Theorem 1 (Böhm)

Let s and t be closed normal λ-terms that are not $\beta\eta$-equivalent. Then there exist closed terms $u_1 \dots u_k$ such that
$(s)u_1 \dots u_k = \lambda xy.y$
$(t)u_1 \dots u_k = \lambda xy.x$

- This means that s and t can be distinguished by their computational behaviour.

Corollary 1

Let $\equiv_\tau$ be an equivalence relation on Λ, containing $\equiv_\beta$, and such that it is λ-compatible. If there exist two normalizable non $\beta\eta$-equivalent terms t, t' such that $t \equiv_\tau t'$, then $v \equiv_\tau v'$ for all terms v, v'.

- The adjunction of another equivalence relation on λ-terms, forces the collapse of the whole set of normal λ-terms.

Maximality of $\equiv_{\beta\eta}$

- The answer is no. It is a consequence of **Böhm's Theorem**.

Theorem 1 (Böhm)

Let s and t be closed normal λ-terms that are not $\beta\eta$-equivalent. Then there exist closed terms $u_1 \dots u_k$ such that
$(s)u_1 \dots u_k = \lambda xy.y$
$(t)u_1 \dots u_k = \lambda xy.x$

- This means that s and t can be distinguished by their computational behaviour.

Corollary 1

Let $\equiv_\tau$ be an equivalence relation on Λ, containing $\equiv_\beta$, and such that it is λ-compatible. If there exist two normalizable non $\beta\eta$-equivalent terms t, t' such that $t \equiv_\tau t'$, then $v \equiv_\tau v'$ for all terms v, v'.

The corollary suggests to take $\beta\eta$-equivalence as a sufficient condition for being a logical constant.

Comparisons with other conditions (1)

A comparison with Belnap's criterion

- η-expansion is more "liberal" than the requirement of *unicity* (Belnap 1961).

- For example, the S4 $\Box$ operator satisfies η-expansion, while it does not satisfies unicity:

η-expansion

$$\cfrac{\cfrac{\cfrac{}{\Box A \vdash \Box A}\,\text{Ax}}{\Box A \vdash A}\,\Box\text{-elim}}{\Box A \vdash \Box A}\,\Box\text{-intro}$$

Unicity

Given two operators, $\Box$ and $\Box^*$, governed by the same rules, we can't prove $\Box A \dashv\vdash \Box^* A$:

$$\cfrac{\cfrac{\cfrac{}{\Box A \vdash \Box A}\,\text{Ax}}{\Box A \vdash A}\,\Box\text{-elim}}{?} \qquad \cfrac{\cfrac{\cfrac{}{\Box^* A \vdash \Box^* A}\,\text{Ax}}{\Box^* A \vdash A}\,\Box^*\text{-elim}}{?}$$

- Therefore Belnap's criterion removes S4 $\Box$ out of the domain of logical constants, while η-expansion does not.

Comparisons with other conditions (2)

A comparison with the general elimination principle

- Consider the quantum disjunction operator:

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \odot B}\ \odot\text{-intro}_1 \qquad\qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \odot B}\ \odot\text{-intro}_2$$

$$\frac{\Gamma \vdash A \odot B \qquad A \vdash C \qquad B \vdash C}{\Gamma \vdash C}\ \odot\text{-elim}$$

  (Where the arbitrary context of the two minor premisses of ⊙-elim must be empty)

- η-expansion condition does not rule out ⊙:

$$\frac{\dfrac{}{A \odot B \vdash A \odot B}\ \text{Ax} \qquad \dfrac{\dfrac{}{A \vdash A}\ \text{Ax}}{A \vdash A \odot B}\ \odot\text{-intro}_1 \qquad \dfrac{\dfrac{}{B \vdash B}\ \text{Ax}}{B \vdash A \odot B}\ \odot\text{-intro}_2}{A \odot B \vdash A \odot B}\ \odot\text{-elim}$$

- On the other hand, given the introduction rules for ⊙, the general inversion principle, without the support of any further condition, "generates" the *usual* rule for disjunction elimination and not the ⊙-elim rule.
- The problem concerns how to impose a control on contexts.

Problems

- Even if we started from computational considerations, then our analysis has been performed only at the linguistic (i.e. of types) level.
- In this manner we risk to persist in the confusion between the meaning-level and the logicality-level.
- Our problem can be rephrased in the following manner: how can be defined an operator in purely λ-terms, without passing through types in advance?
- λ-calculus is a syntactical framework. In order to consider more constructions, we need to extend our definitions of objects and (therefore) of reduction which in this case cannot be considered as primitive.
- If we want to move away from the linguistic level and fully develop the idea that logical constants coincide with those operators that have a particular type of computational behavior, we have then to choose a different setting.
- We want to work in a framework where reduction is defined as a primitive, and where the distinction between logical and non logical constructions on types can be made based on reduction.

Outline

Overview

- The computational level is taken as primitive.
- The leading idea is that the basic computational properties (of programs) are:
  1. Composition / Execution;
  2. Termination.
- Given a sets of “objects” (mathematical objects), a notion of execution and a notion of termination allows one to construct types, as in lambda-calculus (realisability).

Construction

framework	execution	termination
lambda-calculus	β-reduction	(strong) normalizability
permutations	paths	no "internal cycles"
ludics	normalization	daimon
GoI	execution	$\perp = \{0 \cdot + \cdot 1 + 0\}^{\perp}$

- From the notions of execution and termination, we can define a notion of orthogonality.
- From this notion of orthogonality, we can define *types* as sets of objects T such that there exists a set S with $T = S^{\perp}$.

Remark 1

We usually rephrase the definition of a type by saying that a type is a set of objects T such that $T^{\perp\perp} = T$, a statement that is equivalent to the other one.

Remark 2

This allows an object to have multiple types.

MLL sequent calculus

$$\frac{}{\vdash A, A^\perp}\,\text{Ax} \qquad \frac{\vdash \Delta, A \qquad \vdash A^\perp, \Gamma}{\vdash \Delta, \Gamma}\,\text{Cut}$$

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

Proof Structures

Definition 1

A proof structure for MLL is a graph constructed using the following nodes.

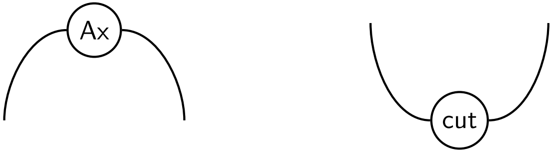


Figure: Liens axiomes et coupures

Proof Structures

Definition 1

A proof structure for MLL is a graph constructed using the following nodes.

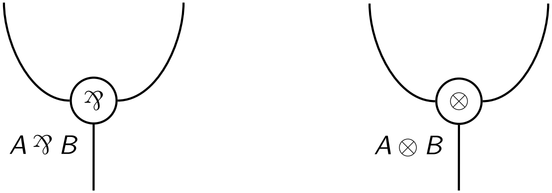


Figure: Liens ⅋ et $\otimes$

Non sequentialisable proofs

Remark 3

Proof structures do not always correspond to a sequent calculus proof.

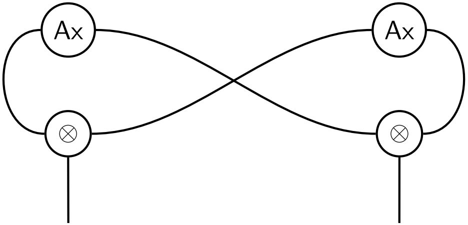


Figure: An example of a non sequentialisable proof

Remark 4

The key point here is the possibility of writing things that are not proofs (as in the syntactic proof of completeness for LK, it gives the syntax a semantical flavour), but we will be able to distinguish the "real" proofs.

Correctness criterions

In order to distinguish sequentialisable proof structures, we use *correctness criterions* : Long trips (LT), Danos-Regnier (DR), counter-proofs (CP), etc.

Correctness criterions

In order to distinguish sequentialisable proof structures, we use *correctness criterions* : Long trips (LT), Danos-Regnier (DR), counter-proofs (CP), etc. Correctness criterion have the same global structure. Let $\mathcal{R}$ be a proof structure:

- We define a family T of objects $\mathcal{R}$: trips (LT), graphs (DR), partitions of a set (CP);
- We show that $\mathcal{R}$ is sequentialisable if and only if each element of T satisfy a given property P : being a long-trip (LT), being connected and acyclic (DR), etc.

Correctness criterions

Looking into the criterions a little further, one can notice that:

- the elements of T are defined only by the *logical part* of the proof structure, i.e. the structure without its axiom links;
- the property P is then a condition on how the axioms interact with the *tests* in T.

Slogan 1

Set of axioms = An untyped proof
Set of tests T = Logical part = Type

Correctness criterions

Looking into the criterions a little further, one can notice that:

- the elements of T are defined only by the *logical part* of the proof structure, i.e. the structure without its axiom links;
- the property P is then a condition on how the axioms interact with the *tests* in T.

Slogan 1

Set of axioms = An untyped proof
Set of tests T = Logical part = Type

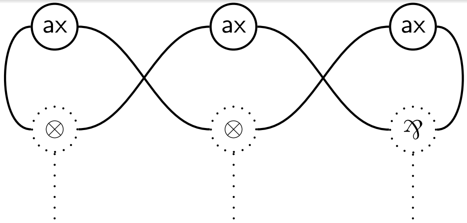
Correctness criterions

Looking into the criterions a little further, one can notice that:

- the elements of T are defined only by the *logical part* of the proof structure, i.e. the structure without its axiom links;
- the property P is then a condition on how the axioms interact with the *tests* in T.

Slogan 1

Set of axioms = An untyped proof
Set of tests T = Logical part = Type



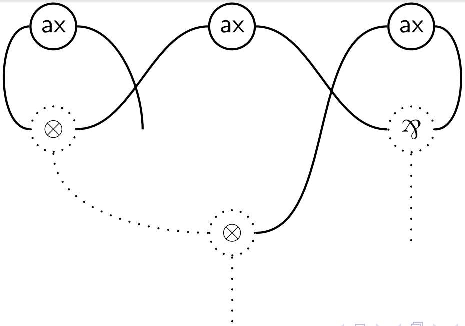
Correctness criterions

Looking into the criterions a little further, one can notice that:

- the elements of T are defined only by the *logical part* of the proof structure, i.e. the structure without its axiom links;
- the property P is then a condition on how the axioms interact with the *tests* in T.

Slogan 1

Set of axioms = An untyped proof
Set of tests T = Logical part = Type



Correctness criterions

Looking into the criterions a little further, one can notice that:

- the elements of T are defined only by the *logical part* of the proof structure, i.e. the structure without its axiom links;
- the property P is then a condition on how the axioms interact with the *tests* in T.

Slogan 1

Set of axioms = An untyped proof
Set of tests T = Logical part = Type

One criterion is particularly interesting, since elements of T and the axiom part both yield permutations. This homogeneity allows us to take one step further: consider elements of T as a kind of proofs (incorrect proofs).

A toy example: permutations

We will show on a simple example how one constructs such a framework.

Definition 2 (Untyped proof)

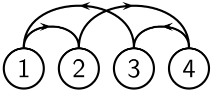
An untyped proof *is a pair* $\mathfrak{a} = \langle X, \sigma \rangle$*, where:*

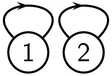
1. $X \in \wp_f(\mathbb{N}) \setminus \{\emptyset\}$ *is called the* location *of* $\mathfrak{a}$*;*
2. σ *is a permutation on* X*.*

Composition

- Let's consider two untyped proofs.

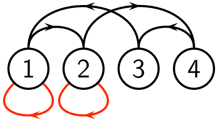
  For exemple, $\mathfrak{a} = \langle \{1,2,3,4\}, (1,2,4,3) \rangle$ and $\mathfrak{b} = \langle \{1,2\}, \mathit{id} \rangle$:





Composition

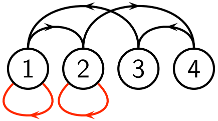
- Their composition is obtained by plugging them together:



- This operation is analogue to the application of a program to another.
- There is a correspondence with the operation of application in pure λ-calculus.

Composition

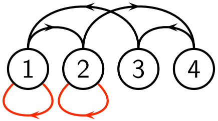
- Their composition is obtained by plugging them together:



- This operation is analogue to the application of a program to another.
- There is a correspondence with the operation of application in pure λ-calculus.

Composition

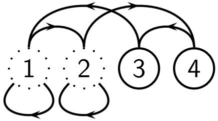
- Their composition is obtained by plugging them together:



- This operation is analogue to the application of a program to another.
- There is a correspondence with the operation of application in pure λ-calculus.

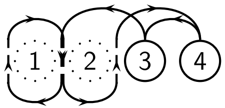
Execution

- The execution of this application gives as a result:



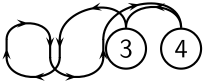
Execution

- The execution of this application gives as a result:



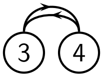
Execution

- The execution of this application gives as a result:



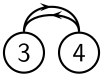
Execution

- The execution of this application gives as a result:



Execution

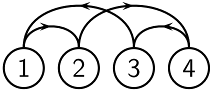
- The execution of this application gives as a result:

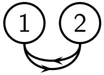


- Execution corresponds to β-reduction in λ-calculus.

Internal Cycles and termination

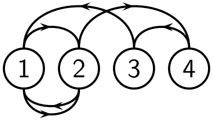
- Sometimes the composition of untyped proofs can generate "internal" cycles (loops).
- For instance, let $\mathfrak{a} = \langle\{1,2,3,4\},(1,2,4,3)\rangle$ and $\mathfrak{b}' = \langle\{1,2\},(1,2)\rangle$:





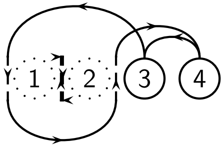
Internal cycles and termination

- The execution of the application yields:



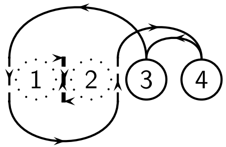
Internal cycles and termination

- The execution of the application yields:



Internal cycles and termination

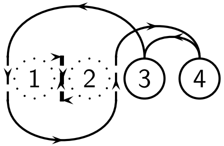
- The execution of the application yields:



- This means that the computation (execution) does not terminate.

Internal cycles and termination

- The execution of the application yields:



- The presence of internal cycles means that the computation does not terminate.
- There is an analogy with non-terminating reductions of pure λ-terms, e.g. $(\lambda x(x)x)\lambda x(x)x$.

Application of untyped proofs

- The genuine application of two untyped proofs can be stated in the following way:

Definition 3 (Application)

Let be $\mathfrak{a} = \langle X \cup Y, \sigma \rangle$ and $\mathfrak{b} = \langle Y, \tau \rangle$, with $X \cap Y = \emptyset$ and let π_X be the partial identity on X.
The application of $\mathfrak{a}$ to $\mathfrak{b}$ is defined when no internal cycles appear and it is then defined as:

$$[\mathfrak{a}]\mathfrak{b} = \langle X, \sigma \dagger \tau \rangle$$

where

$$\sigma \dagger \tau = \pi_X(\sigma \cup \sigma\tau\sigma \cup \sigma\tau\sigma\tau\sigma \ldots)\pi_X$$

Orthogonality

Definition 4 (Orthogonality)

Two untyped proofs $\mathfrak{a} = \langle X, \sigma \rangle$ *and* $\mathfrak{b} = \langle X, \tau \rangle$ *are* orthogonal *if and only if* $\sigma\tau$ *is a cyclic permutation.*

- This intuitively means that a program $\mathfrak{a}$ is tested ("confronted") with another one $\mathfrak{b}$ and that $\mathfrak{b}$ is "accepted" by $\mathfrak{a}$ and *vice versa* (i.e. $\mathfrak{a}$ pass the test of $\mathfrak{b}$ and $\mathfrak{b}$ pass the test of $\mathfrak{a}$).
- Note that the condition for the orthogonality of two untyped proofs represents a special case of a terminating execution (this is more obvious in Ludics or GoI).

From untyped proofs to types

Definition 5

A type is a set of untyped proofs T such that there exists a set S of untyped proofs with $T = S^{\perp\!\!\!\perp} = \{\sigma \mid \sigma \perp\!\!\!\perp \tau, \forall \tau \in S\}$.

We already pointed out that it s equivalent to:

Definition 6 (Type)

A subset $\mathbf{A}$ of $\mathfrak{S}(X)$ equal to its bi-orthogonal $\mathbf{A}^{\perp\!\!\!\perp\perp\!\!\!\perp}$ is called a type *(of carrier X).*

Intuitively, this means that if two untyped proofs are in the same type, they somewhat behave in the same way (since they both pass a given set of tests).

Logical operations (1)

- In this setting we can define multiplicative connectives of linear logic.
- Let $\mathfrak{a} = \langle X, \sigma \rangle$ and $\mathfrak{b} = \langle Y, \tau \rangle$, where $X \cap Y = \emptyset$. We can define the *tensor product* of $\mathfrak{a}$ and $\mathfrak{b}$ by:

$$\mathfrak{a} \otimes \mathfrak{b} = \langle X \cup Y, \sigma \cup \tau \rangle$$

- Let **A** and **B** two types of respective carriers X and Y, where $X \cap Y = \emptyset$. We define the type $\mathbf{A} \otimes \mathbf{B}$ of carrier $X \cup Y$ by:

$$\mathbf{A} \otimes \mathbf{B} = \{\mathfrak{a} \otimes \mathfrak{b} \,|\, \mathfrak{a} \in \mathbf{A} \text{ and } \mathfrak{b} \in \mathbf{B}\}^{\perp\perp}$$

- The operator $\otimes$ satisfies the following properties:
  - $\mathbf{A} \otimes \mathbf{B} = \mathbf{B} \otimes \mathbf{A}$ (Commutativity)
  - $\mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C}) = (\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C}$ (Associativity)

Logical operations (2)

- Let be $\mathbf{A}$ and $\mathbf{B}$ two types and consider the set

$$\mathbf{A} \multimap \mathbf{B} = \{\mathfrak{f} \mid \forall \mathfrak{a} \in \mathbf{A}, [\mathfrak{f}]\mathfrak{a} \in \mathbf{B}\}$$

Theorem 2

The following equivalence holds: $\mathbf{A} \multimap \mathbf{B} = (\mathbf{A} \otimes \mathbf{B}^{\perp\!\!\!\!\smile})^{\perp\!\!\!\!\smile}$.

Corollary 2

The set $\mathbf{A} \multimap \mathbf{B}$ *is a type.*

Truth

In such frameworks, we can define a notion of correct proofs and truth.

Definition 7

An untyped proof $\langle X, \sigma \rangle$ is correct when it is a disjoint union of transpositions, i.e. when $\sigma^2 = Id$ and $\sigma(x) \neq x$ for all $x \in X$.

Definition 8

A type is true when it contains a correct proof.

Proposition 1

Truth is preserved by the $\otimes$ operation and application (execution).

Logical and non-logical operations

- Every operation on untyped proofs allows one to define an operation on types. For instance, if $g(x, y)$ defines an untyped proofs from two untyped proofs x and y, we define the operation on types

$$g(A, B) = \{g(x, y) : x \in A, y \in B\}^{\perp\perp}$$

- Given a type it is also always possible to define its dual at the level of types. However, this dual cannot always be expressed through an operation over untyped proofs.

An operation on types will be a logical constant when both it and its dual have a computational meaning, i.e. when they are defined as a "natural" construction on untyped proofs.

- For instance, in the permutations framework a natural construction on untyped proofs can be defined as a construction that preserves inclusions and/or correctness.

Non-logical operations: An example

- Let's take an untyped proof $\mathfrak{a} = \langle X, \sigma \rangle$ and define the operation of *square* exponentiation:
$$\mathfrak{a}^2 = \langle X, \sigma^2 \rangle$$
- Given a type **A**, the square operation over untyped proofs induces the new type:
$$\boxdot \mathbf{A} = \{\mathfrak{a}^2 \mid \mathfrak{a} \in A\}^{\perp\perp}$$

We now take a look at some examples:

- The sets $\mathbf{F}_2 = \{\langle \{1,2\}, id \rangle\}$ and $\mathbf{F}_3 = \{\langle \{1,2,3\}, id \rangle\}$ are types.
- We have
$$\mathbf{C}_2 = \{\langle \{1,2\}, (1,2) \rangle\} = \mathbf{F}_2^{\perp}$$
$$\mathbf{C}_3 = \{\langle \{1,2,3\}, (1,2,3) \rangle, \langle \{1,2,3\}, (1,3,2) \rangle\} = \mathbf{F}_3^{\perp}$$
- These equalities are satisfied: $\boxdot \mathbf{F}_2 = \mathbf{F}_2$, $\boxdot \mathbf{C}_2 = \mathbf{F}_2$, $\boxdot \mathbf{F}_3 = \mathbf{F}_3$, $\boxdot \mathbf{C}_3 = \mathbf{C}_3$, hence $(\boxdot \mathbf{C}_2)^{\perp} = \mathbf{C}_2$ and $(\boxdot \mathbf{C}_3)^{\perp} = \mathbf{F}_3$

Advantages of these frameworks

- The framework is rich enough: each operation on untyped proofs defines an operation on types and there already exists many properties of untyped proofs and types that could be used to distinguish between logical and non logical operations on types, for instance:
  - the preservation of correctness;
  - the "naturality", i.e. the preservation of inclusion in case of permutations;
  - internal completeness (i.e. the closure by bi-orthogonality is not necessary);
  - ...
- The notion of execution needs not be adapted when a new construction is introduced, as opposed to what happens when one works with lamda-calculus.

Outline

By way of conclusion (1)

- In the first two parts we have analyzed a series of principles that give sufficient conditions for being a logical constant.
- A linguistic operator (i.e. an operator applied over types) is *ruled out* of the domain of logical constants if it does not respect those principles.
- Instead, by taking as primitive the operations over untyped objects what we get is a condition for allowing an operator to *enter* into the domain of the logical constants.
- What about inferentialism?
- Strictly speaking, our computational untyped setting cannot be considered as an inferentialist account.

By way of conclusion (2)

- In the *Tractatus*, Wittgenstein points out

All inference is made a priori (§5.133)

- TUP makes explicit the interplay between the *a priori* rules of a logical setting and the *a posteriori* normativity (Girard) of the untyped setting.

- TUP as an interactional framework (execution and orthogonality have a "dialogical flavor")

  From a philosophical point of view UPT can be seen as a useful analytical tool which allows the comparison between different approaches to the meaning of logical constants:

  - Dummett/Prawitz - Harmony for verificationist/pragmatist theories
  - Martin-Löf - Curry-Howard and judgemental methods
  - Brandom - Normativity and intentionality
  - ...