# Logical Constants from a Computational Point of View

Alberto Naibo[1] Mattia Petrolo[2]

[1]Université Paris 1 - EXeCO

[2]Université Paris 7 - Università Roma Tre

March 22, 2010
Workshop PROOFS AND MEANING
MSH, Paris

# Logical Constants from a Computational Point of View

Alberto Naibo[1] Mattia Petrolo[2]

[1]Université Paris 1 - EXeCO

[2]Université Paris 7 - Università Roma Tre

March 22, 2010
Workshop PROOFS AND MEANING
MSH, Paris

# Outline


1. Introduction
2. Proofs-as-programs correspondence
3. Applications of $\equiv_{\beta\eta}$ to the theory of meaning
4. Conclusions and future work

# Outline

1. Introduction

2. Proofs-as-programs correspondence

3. Applications of $\equiv_{\beta\eta}$ to the theory of meaning

4. Conclusions and future work

# Logical inferentialism (1)

## 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 assign 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])

- $\mathtt{tonk}$ connective shows that some constraints are needed in order to define correctly the meaning of logical constants.

$$
\dfrac{
\dfrac{
\Rightarrow -intro\;\dfrac{\mathtt{tonk}-elim_2\;\dfrac{\mathtt{tonk}-intro_1\;\dfrac{Ax\;\dfrac{}{A \vdash A}}{A \vdash A\,\mathtt{tonk}\,B}}{A \vdash B}}{\vdash A \Rightarrow B}
\qquad
\dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash B}\;Ax}{B \vdash A\,\mathtt{tonk}\,B}\;\mathtt{tonk}-intro_2}{B \vdash A}\;\mathtt{tonk}-elim_1}{\vdash B \Rightarrow A}\;\Rightarrow -intro
}{\vdash (A \Rightarrow B) \wedge (B \Rightarrow A)}\;\wedge - intro
}{\vdash A \Leftrightarrow B}
$$

# Logical inferentialism (2)

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* (Prawitz [1973], Dummett [1991]): 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 criterion bans tonk

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

- It is impossible to define a normalization strategy.

# A problem with *harmony-as-normalization* (1)

- The criterion of *harmony-as-normalization* does not ban all the pathological constants: ≪harmony is an excessively modest demand≫ (Dummett [1991], p. 287).

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

$$\star\text{−}intro\ \frac{\Gamma \vdash A \qquad \Gamma' \vdash B}{\Gamma, \Gamma' \vdash A \star B} \qquad\qquad \frac{\Gamma \vdash A \star B \qquad \Gamma' \vdash A}{\Gamma, \Gamma' \vdash B}\ \star\text{−}elim$$

- These rules enjoy a normalization strategy:

$$\star\text{−}elim\ \dfrac{\star\text{−}intro\ \dfrac{\dfrac{\mathcal{D}}{\Gamma \vdash A} \qquad \dfrac{\mathcal{D}'}{\Gamma' \vdash B}}{\Gamma, \Gamma' \vdash A \star B} \qquad \dfrac{\mathcal{D}''}{\Gamma'' \vdash A}}{\Gamma, \Gamma', \Gamma'' \vdash B} \quad \rightsquigarrow \quad \dfrac{\mathcal{D}'}{\Gamma' \vdash B}$$

# A problem with *harmony-as-normalization* (2)

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

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

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

$$\dfrac{\dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}\,Ax}{A \wedge B \vdash A}\wedge - elim_1 \qquad \dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}\,Ax}{A \wedge B \vdash B}\wedge - elim_2}{A \wedge B \vdash A \wedge B}\wedge - intro$$

- This procedure fails for ✶:

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

# A problem with *harmony-as-normalization* (3)

- 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* (4)

- Therefore, the failure of deducibility of identicals is a sign that something is wrong with $\ast$.
- Why is this property important? How can we justify it?
- In order to answer these questions, let us look at the *computational* properties of Natural Deduction.

# Outline

1. Introduction
2. Proofs-as-programs correspondence
3. Applications of $\equiv_{\beta\eta}$ to the theory of meaning
4. Conclusions and future work

# Curry-Howard isomorphism

- The Curry-Howard isomorphism establishes a one-to-one correspondance between Natural Deduction and $\lambda$-calculus, e.g.

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

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

  Indeed, *normalization* in NJ corresponds to $\beta$-reduction in $\lambda$-calculus.

- $\lambda$-terms are considered as programs and a type judgement $t : A$ is called a program specification.

- The $\beta$-reduction correponds to a program execution, i.e. the *computation* of a certain program.

# $\eta$-expansion

- In $\lambda$-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 ($\eta$-*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 $\eta$-expansion is type-preserving.

# $\eta$-expansion and deducibility of identicals

- $\eta$-expansion corresponds exactly to the property of deducibility of identicals:

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

$$\dfrac{\dfrac{\dfrac{}{t : A \wedge B \vdash t : A \wedge B}\,Ax}{t : A \wedge B \vdash \pi_1(t) : A}\wedge - elim_1 \qquad \dfrac{\dfrac{}{t : A \wedge B \vdash t : A \wedge B}\,Ax}{t : A \wedge B \vdash \pi_2(t) : B}\wedge - elim_2}{t : A \wedge B \vdash \langle \pi_1(t), \pi_2(t)\rangle : A \wedge B}\wedge - intro$$

# Extensionality in $\lambda$-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 demand to add some other type-preserving relation on $\lambda$-terms?

# Maximality of $=_{\beta\eta}$

- The answer is no. It is a consequence of **Böhm's Theorem**, i.e.
- **Theorem**. *Let $s$ and $t$ be closed normal $\lambda$-terms that are not $\beta\eta$-equivalent. Then there exist closed terms $u_1 \ldots u_k$ such that*

  $$(s)u_1 \ldots u_k = \lambda xy.y$$
  $$(t)u_1 \ldots u_k = \lambda xy.x$$

  i.e. $s$ and $t$ can be distinguished by their computational behaviour.
- **Corollary**. Let $\equiv_\tau$ be an equivalence relation on $\Lambda$, containing $\equiv_\beta$, and such that it is $\lambda$-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 $\lambda$-terms, forces the collapse of the whole set of normal $\lambda$-terms.

# Maximality of $=_{\beta\eta}$

- The answer is no. It is a consequence of **Böhm's Theorem**, i.e.
- **Theorem**. *Let $s$ and $t$ be closed normal $\lambda$-terms that are not $\beta\eta$-equivalent. Then there exist closed terms $u_1 ... u_k$ such that*

  $$(s)u_1 ... u_k = \lambda xy.y$$
  $$(t)u_1 ... u_k = \lambda xy.x$$

  i.e. $s$ and $t$ can be distinguished by their computational behaviour.
- **Corollary**. Let $\equiv_\tau$ be an equivalence relation on $\Lambda$, containing $\equiv_\beta$, and such that it is $\lambda$-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.

# Outline


1. Introduction
2. Proofs-as-programs correspondence
3. Applications of $\equiv_{\beta\eta}$ to the theory of meaning
4. Conclusions and future work

# $\equiv_{\beta\eta}$ and theory meaning

- The $\equiv_{\beta\eta}$ allows to answer to two fundamental questions in the theory of meaning:

1) when are two different assertions, involving the same proposition, identical?
2) when do two different propositions have the same meaning?

- The first question can be reformulated in the following manners:
  - 1') when are two program specifications identical?
  - 1'') when are we making the same judgment?

- The second question corresponds to the search of a *synonymity criterion* for propositions.

# Identity criteria for assertions

- The computational approach allows to distinguish between two types of identity criterion:

1. **Intentional criterion**: two assertions are intensionally identical iff they are $\beta$-equivalent. To establish their identity is sufficient to take the two $\lambda$-terms, eliminate all their detours (redex) and look if they converge to the same normal form. The procedure is completely internal to the two $\lambda$-terms, no other information is necessary.
2. **Extensional criterion**: two assertions (not $\beta$-equivalent) are extensionally identical iff, when put into the same context, they produce the same effects. This is nothing else that being $\eta$-equivalent.

This last criterion corresponds to a sort of *principle of identity of indiscernibles* formulated for assertions: two assertions are identical when it is not possible to distinguish them on the basis of their behaviour in all possible contexts of application.

- Putting these two criteria together we get a full criterion for identity of assertions:

two assertions are identical iff they are $\beta\eta$-equivalent.

# Example (1)

- In the light of Curry-Howard isomporhism all the $\lambda$-terms of type

$$Nat := (N \Rightarrow N) \Rightarrow (N \Rightarrow N)$$

(with $N$ atomic) correspond to a Church numeral: $\overline{n} = \lambda f \lambda x \underbrace{(f) \ \ldots (f)}_{n} x$

(with $x : N$ and $f : N \Rightarrow N$).

- When we apply the program that corresponds to the sum

$$+ := \lambda m \lambda n \lambda f \lambda x((m)f)((n)f)x : Nat \Rightarrow (Nat \Rightarrow Nat)$$

to two numerals, we obtain a non-normal $\lambda$-term of type $Nat$.
For example, take $m = n = \overline{1} = \lambda f.\lambda x(f)x : Nat$,

$$\overline{1} + \overline{1} = ((\lambda m \lambda n \lambda f \lambda x((m)f)((n)f)x)\lambda f \lambda x(f)x)\lambda f \lambda x(f)x : Nat$$

is a *non-normal* $\lambda$-term that, once $\beta$-reduced, brings to
$\overline{2} = \lambda f \lambda x(f)(f)x : Nat$, which is in normal form.

- By the intensional criterion we can say that the two assertions $\overline{2} : Nat$ and $\overline{1} + \overline{1} : Nat$ are intentionnally identical.

# Example (2)

- Given the two assertions

$$\lambda f.f : (N \Rightarrow N) \Rightarrow (N \Rightarrow N)$$

  and

$$\lambda f \lambda x (f) x : (N \Rightarrow N) \Rightarrow (N \Rightarrow N)$$

  if the only identity criterion was the intentional one, we would be obliged to affirm that the two assertions are different, beacuse they are both in $\beta$-normal form.

- On the other hand, it is easy to check that, when applied to any terms $u : N \Rightarrow N$ and $v : N$, the two assertions give the same result of type $N$:

$$((\lambda f.f)u)t \rightsquigarrow_\beta (u)t$$

$$((\lambda f \lambda x (f) x) u) t \rightsquigarrow_\beta (u)t$$

- The second assertion is just an $\eta$-expansion of the first one.
  It's only with the extensional criterion that we can judge these two assertions as identical (they both stand for $\overline{1} : Nat$).

# Non-identical assertions

- The quotient obtained by the $\beta\eta$-equivalence relation over the class of the $\lambda$-terms of the same type $A$ is not degenerated: not all assertions involving the same proposition $A$ are identified.
- For example,

  $$\lambda z \lambda y \lambda x.x : (A \Rightarrow A) \Rightarrow (B \Rightarrow (A \Rightarrow A))$$

  and

  $$\lambda z \lambda y.z : (A \Rightarrow A) \Rightarrow (B \Rightarrow (A \Rightarrow A))$$

  are not $\beta\eta$-equivalent.
- This means that the two assertions can be justified in different ways.
- Moreover, once they interact with a certain context they behave in different manners and produce different results. In an Austinian sense, we can say that the same proposition can be used to *do* different things.

# A criterion for identity of meaning?

- Is there a relation between two different propositions that allows to identify them with respect to meaning?
- Certainly the logical equivalence relation is not a plausible candidate: it acts only at formulas level, i.e. what counts is just the fact of having the same truth-values in all possible models.
- If we want to respect the inferentialist semantics we have to look for another candidate, namely a relation that acts at proofs level.

# Isomorphism of types

- **Isomorphism of types**: two types $A$ and $B$ are isomorphic iff there exist two morphisms $f : A \rightarrow B$ and $g : B \rightarrow A$, such that $g \circ f = Id_A$ and $f \circ g = Id_B$.

- **Computational isomoprhism ($\lambda$-calculus)**: two types $A$ and $B$ are computationally isomorphic iff there exist two sequents of the form $x : A \vdash t_1 : B$ and $y : B \vdash t_2 : A$ (with $x$ free in $t_1$ and $y$ free in $t_2$) such that for the two $\lambda$-terms $\lambda x.t_1 : A \Rightarrow B$ and $\lambda y.t_2 : B \Rightarrow A$ holds $(\lambda y.t_2)t_1 : A \equiv_{\beta\eta} x : A$ and $(\lambda x.t_1)t_2 : B \equiv_{\beta\eta} y : B$.

Alternatively (working with closed terms): $A$ and $B$ are computationally isomorphic iff $\lambda x(\lambda y.t_2)t_1 : A \Rightarrow A \equiv_{\beta\eta} \lambda x.x : A \Rightarrow A$ and $\lambda y(\lambda x.t_1)t_2 : B \Rightarrow B \equiv_{\beta\eta} \lambda y.y : B \Rightarrow B$.

# Computational isomorphism and logical equivalence

- It is important to note that computational isomorphism *refines* the notion of logical equivalence: the equivalence relation induced by computational isomorphism is strictly stronger than the relation of logical equivalence.

- For example $A \wedge A \dashv\vdash A$, nonetheless it is not a computational isomorphism.

**Note**: in this case, the fact that they are not computationally isomorphic can be appreciated especially at the $\lambda$-terms level:

i. if we compose $x : A \vdash \langle x, x \rangle : A \wedge A$ with $t : A \wedge A \vdash \pi_1(t) : A$, we obtain $t : A \wedge A \vdash (\lambda x.\langle x, x \rangle)\pi_1(t) : A \wedge A$.

ii. After $\beta$-reduction we get: $t : A \wedge A \vdash \langle \pi_1(t), \pi_1(t) \rangle : A \wedge A$.

iii. Now, $\langle \pi_1(t), \pi_1(t) \rangle$ is not a $\eta$-expansion of $t$, so we can't return to the identity $t : A \wedge A \vdash t : A \wedge A$.

# Synonymity criterion

- In the light of the Curry-Howard isomorphism a proposition corresponds to a type.

**Thesis (Došen [2003])**

Two propositions are synonymic iff they are computationally isomorphic.

- Two propositions that are computationnally isomorphic behave in the same manner in proofs:

Given two isomporhic propositions $A$ and $B$, if there is a proof $\alpha$ in which one of them, say $A$, figure as assumption (resp. as conlcusion), it is possible to compose $\alpha$ with a proof $\beta$ of $B \vdash A$ (resp. $A \vdash B$), obtaining a proof $\alpha'$ in which $A$ is replaced by $B$, so that nothing is lost, nor gained. Indeed, it is always possible to invert the process and restore the initial situation: by composing $\alpha'$ with a proof $\gamma$ of $A \vdash B$ (resp. $B \vdash A$), we obtain, after $\beta\eta$-reduction, the original proof $\alpha$.

- This means that $A$ and $B$ are mutually interchangeable in proofs and that the computational effects of this operation can always be annulled.

# An exemple

Given the proof $\dfrac{\dfrac{\dfrac{A \wedge B}{A}\,\wedge\text{-elim}_1}{A \vee B}\,\vee\text{-intro}_2}{D \Rightarrow (A \vee B)}\,\Rightarrow\text{-intro}$ and the isomorphic propositions $A \wedge B$ and $B \wedge A$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[A \wedge B]^1}{A}\,\wedge\text{-elim}_1}{A \vee B}\,\vee\text{-intro}_2}{D \Rightarrow (A \vee B)}\,\Rightarrow\text{-intro}}{(A \wedge B) \Rightarrow (D \Rightarrow (A \vee B))}\,\Rightarrow\text{-intro (1)} \quad \dfrac{\dfrac{B \wedge A}{A}\,\wedge\text{-elim}_2 \quad \dfrac{B \wedge A}{B}\,\wedge\text{-elim}_1}{A \wedge B}\,\wedge\text{-intro}}{D \Rightarrow (A \vee B)}\,\Rightarrow\text{-elim} \leadsto \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{B \wedge A}{A}\,\wedge\text{-elim}_2 \quad \dfrac{B \wedge A}{B}\,\wedge\text{-elim}_1}{A \wedge B}\,\wedge\text{-intro}}{A}\,\wedge\text{-elim}_1}{A \vee B}\,\vee\text{-intro}_2}{D \Rightarrow (A \vee B)}\,\Rightarrow\text{-intro}$$

$$\leadsto \dfrac{\dfrac{\dfrac{B \wedge A}{A}\,\wedge\text{-elim}_2}{A \vee B}\,\vee\text{-intro}_2}{D \Rightarrow (A \vee B)}\,\Rightarrow\text{-intro} \leadsto \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[B \wedge A]^1}{A}\,\wedge\text{-elim}_2}{A \vee B}\,\vee\text{-intro}_2}{D \Rightarrow (A \vee B)}\,\Rightarrow\text{-intro}}{(B \wedge A) \Rightarrow (D \Rightarrow (A \vee B))}\,\Rightarrow\text{-intro (1)} \quad \dfrac{\dfrac{A \wedge B}{B}\,\wedge\text{-elim}_2 \quad \dfrac{A \wedge B}{A}\,\wedge\text{-elim}_1}{B \wedge A}\,\wedge\text{-intro}}{D \Rightarrow (A \vee B)}\,\Rightarrow\text{-elim}$$

$$\leadsto \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A \wedge B}{B}\,\wedge\text{-elim}_2 \quad \dfrac{A \wedge B}{A}\,\wedge\text{-elim}_1}{B \wedge A}\,\wedge\text{-intro}}{A}\,\wedge\text{-elim}_2}{A \vee B}\,\vee\text{-intro}_2}{D \Rightarrow (A \vee B)}\,\Rightarrow\text{-intro} \leadsto \dfrac{\dfrac{\dfrac{A \wedge B}{A}\,\wedge\text{-elim}_1}{A \vee B}\,\vee\text{-intro}_2}{D \Rightarrow (A \vee B)}\,\Rightarrow\text{-intro}$$

# Outline

1. Introduction
2. Proofs-as-programs correspondence
3. Applications of $\equiv_{\beta\eta}$ to the theory of meaning
4. Conclusions and future work

# Conclusions

- The properties of $\beta$-reduction and $\eta$-expansion (or, more generally $\equiv_{\beta\eta}$) are the minimal requirements that we demand for 'defining' a logical constant.
- The advantage of working in a computational setting is that it is easier to detect some essential proof-theoretical properties. An example is the $\eta$-expansion: it is a property that naturally emerges in the characterization of a consistent computational system, but that it is more difficult to justify in purely logical terms.
- Moreover, our approach constitutes an attempt to solve some basic questions of a theory of meaning, such as the identity criterion for assertions and the problem of the notion of synonymity.

# Future work

- A comparison of our criterion with Dummett's *stability* (Dummett [1991]) and Negri/von Plato's *general inversion principle* (Negri/von Plato [2001], Negri [2002]).

  For exemple, can we establish a hierarchy between these criteria (from the weakest to the strongest)?