On the Computational Meaning of Axioms

Alberto Naibo, Mattia Petrolo, Thomas Seiller

On the Computational Meaning of Axioms

Alberto Naibo[✧], Mattia Petrolo[✧], and Thomas Seiller[☆]

[✧]*Institut d'Histoire et de Philosophie des Sciences et des Techniques (IHPST), Université Paris 1 Panthéon-Sorbonne – ENS – CNRS (UMR 8590), 13, rue du Four, 75006 Paris, France*
[☆]*Institut des Hautes Études Scientifiques (IHÉS), Le Bois-Marie 35, route de Chartres, 91440 Bures-sur-Yvette, France*

Abstract

An anti-realist theory of meaning suitable for both logical and proper axioms is investigated. As opposed to other anti-realist accounts, like Dummett-Prawitz verificationism, the standard framework of classical logic is not called into question. In particular, semantical features are not limited solely to inferential ones, but also computational aspects play an essential role in the process of determination of meaning. In order to deal with such computational aspects, a relaxation of syntax is shown to be necessary. This leads to a general kind of proof theory, where the objects of study are not typed objects like deductions, but rather untyped ones, in which formulas have been replaced by geometrical configurations.

1 Introduction

The standard conception of axiom is characterized by a conceptual and ontological priority assigned to the notion of *structure*. Starting from a certain «body of facts [*Tatsachenmaterial*]» (see Hilbert 1905, translated in Hallett 1995, p. 136) composed by propositions, theorems, conjectures, and proof methods belonging to different mathematical systems, it is possible to single out some *invariants* that allow to identify the common features of these systems. In this process of *abstraction* a general and univocal form is pointed out and the axioms fix, at the linguistic level, this abstract form that «might be called a relational structure» (Bernays 1967, p. 497). When formalized in a set-theoretical way, this notion of structure becomes ontologically concrete and it can play the role of an *interpretation structure*, i.e. a model, both for the axioms and for the other sentences derivable from the axioms. In this sense, we can say that the grounding idea of the axiomatic method is to capture a class of models sharing some relevant properties that distinguish them from other classes of models. An immediate

E-mail addresses: alberto.naibo@univ-paris1.fr (A. Naibo), mattia.petrolo@univ-paris-diderot.fr (M. Petrolo), seiller@ihes.fr (T. Seiller).

consequence is that the proper axioms of a certain theory T are considered as meaningful because they are *true* exactly in those classes of models that they identify. It seems then that the notion of axiom fits well with a truth-conditional, or model-based, theory of meaning (see Naibo 2013, ch. 3). It is in this direction that goes, for instance, Hintikka's remark that the genuine relation between axioms and theorems is the model-theoretical relation of logical consequence rather than the syntactical relation of derivability (Hintikka 2011, pp. 73-75). Derivations are subordinated to semantical aspects, in the sense that their role is reduced to guarantee truth transmission. This idea finds a further confirmation in the difficulty of constructing an inferentialist theory of meaning – for example, in the style of Dummett-Prawitz verificationism – when it has to deal with axioms. In particular, in presence of proper axioms the fundamental notion of *canonical proof* is lost. Consider, for example, the derivation in natural deduction of the sentence $\forall x(x = 0 \vee \exists y(x = s(y)))$ from Peano's axioms (no matter that we are using classical or intuitionistic inference rules). The derivation terminates with an application of the $\rightarrow$ elimination rule having as major premiss an instance of the axiom scheme of induction. This is not a *canonical proof* in the sense of the (inferential) verificationism of Dummett-Prawitz, since a canonical proof is a proof terminating with the introduction rule of the principal connective of the sentence under analysis – in this case the $\forall$ introduction rule.[1] This means that, in general, in presence of proper axioms it is not possible to reduce to a common form – or to identify a common feature of – all possible proofs of those sentences having the same principal connective. The immediate consequence is that the notion of proof cannot be used for explaining the meaning of sentences: in absence of a common form to which they can be reduced, different proofs of the same sentence would turn out to confer different meanings to it, so that at last we cannot even properly refer to it as the *same sentence*.[2] This would be particularly problematic for mathematics, where a theorem is supposed to possess always the same meaning, even if proved in different ways.

But inferentialist theories of meaning have a further characteristic feature: they are closely related to anti-realist positions according to which semantical concepts have not to transcend our epistemic capacities. Now, if in logic and mathematics we do not want to abandon this epistemic-based perspective, then we have not to abandon the semantical key concept of canonical proof, since a canonical proof is a finite object the nature of which does not go beyond our epistemic capacities. In order not to abandon this key concept, it seems that the only possible solution is to give up the notion of axiom in favor of other alternative notions, namely that of non-logical rule of inference (Negri & von Plato 1998) or that of rewrite rule (Dowek et al. 2003). However, from a philosophical

[1] Notice that here we prefer to exclude from the set of canonical proofs those that are *trivially canonical*, i.e. those proofs terminating with a sequence of c-elim/c-intro rules, with c as the principal connective of the conclusion. Moreover, it should be mentioned that another well known example of axiomatic theories that prevent form the possibility of obtaining canonical proofs, namely of canonical proofs of disjunctive or existential sentences, is represented by those theory the axioms of which contain strictly positive occurrences of disjunctive and existential sentences (see Troelstra & Schwichtenberg 2000, pp. 6, 106-107).

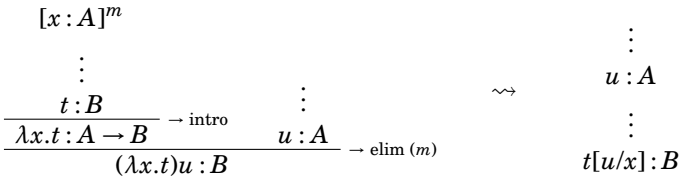
[2] This position is usually identified with a Wittgensteinian position (see Wittgenstein 1956, Part II, §31, Part V, §7), nevertheless we think that this is a mistake. The point is that Wittgenstein is not speaking of the meaning of a sentence considering it as something abstract, invariant and objective – as the propositional content of that sentence could be – but he is speaking instead of how each single agent gets to a specific understanding of that sentence, depending on the particular place she assigns to it inside her own web of beliefs.

point of view, this way of proceeding eventually leads to substantial revisionist positions. Indeed, embracing an inferentialist and anti-realist theory of meaning not only brings to adopt revisionist positions with respect to logical constants (cf. Dummett 1973; Dummett 1991, pp. 291-300) – namely by abandoning classical operators for intuitionistic ones – but it induces also to be revisionist with respect to the usual and commonly accepted image of mathematical theories – namely by abandoning the standard conception of a theory as a set of axioms and replacing it with the conception of a theory as a set of postulates (i.e. hypothetical actions or *Erzeugungsprinzipien*; see for example von Plato 2007, p. 199) or as a an algorithm (i.e. a set of computational instructions, see Dowek 2010). These solutions are analyzed in details in Naibo (2013, Part III).

What we propose in this paper is instead a way to save a theory of meaning essentially based on the notion of inference and proof, but without necessarily abandon the notion of axiom. In order to do that, we will adopt what can be called an *interactional* point of view. More precisely, differently from standard Dummettian inferentialism, we will enlarge our set of semantic key concepts, accepting not only objects containing exclusively correct instantiations of axioms and rules – as it is the case for canonical proofs –, but also objects containing incorrect ones. For this reason, they will be called *paraproofs* and, differently from proofs, they will be essentially *untyped* objects. This means that types – i.e. propositions or sentences – are no more the primitive entities on which inference rules act on, but they become the outcome of the interaction between paraproofs. A quite natural setting where to model this notion of interaction is the *computational* one and especially the so-called Curry-Howard correspondence will be our starting point (see Sørensen & Urzyczyn 2006 for a comprehensive presentation). From such of a perspective, a formal correlation between proofs and programs is established; in particular, proofs can be seen as the surface linguistic "description" of the inner intensional behavior of programs. Our idea is then to show how it is possible to construct semantical aspects starting from the behavior of programs. More precisely, our aim is to show that to know the meaning of an axiom does not consist in knowing which objects and which structures render it true, but in knowing which is the computational behavior of the program associated to this axiom, once it has been put in interaction with other programs.

2 Axioms and computation

From an historical point of view, the proofs-as-programs correspondence was, at first, stated between (deductive systems for) constructive logics and abstract functional programming languages, in particular between minimal or intuitionistic natural deduction and λ-calculus. In this setting, the execution of a program – i.e. a λ-term – having specification A corresponds to the normalization of a natural deduction proof having A as conclusion. More precisely, each (local) elimination of a *detour* of the form c-intro/c-elim – where c is a logical connective – corresponds to a step of program execution. For example, the elimination of a $\rightarrow$ *detour* corresponds to the execution of one step of β-reduction, that is one step of the computation of the value taken by the function/program $\lambda x.t$ once the latter is applied to the input u:



Computation seems then to be necessarily tied to non-atomic – i.e. complex – types, namely the maximal formulas of the *detours*.[3] In this respect, it is worth noting that there exists two types of formulas that can never appear as maximal formulas: proper axioms and assumptions.[4] The reason is trivial. Proper axioms and assumptions are always the starting points of derivations, therefore they cannot be preceded by an introduction rule and thus no detour can be created.

Actually, the analysis just sketched can be refined by making appeal to two properties resulting from Prawitz's translation of natural deduction into sequent calculus[5] (Prawitz 1965, pp. 90-91; von Plato 2003, §5). The first property is that *only* detours are translated into instances of the cut rule. In this way cut-formulas coincide with the maximal formulas and thus cuts are always non-atomic.[6] The second one is that the translation operates by transforming proper axioms – or their instances, in the case of axiom schemes – into part of the context of derivations, i.e. it moves proper axioms – or their instances – from the top position of natural deduction derivations into the left handside of sequents.

For example, consider the theory of equality presented by the two axioms

(Ref) $\forall x(x = x)$

(Euc) $\forall x \forall y \forall z(x = y \wedge x = z \rightarrow y = z)$

The non-normal proof in natural deduction shown in Figure 1(a) is translated as the sequent calculus proof shown in Figure 1(b).

The immediate consequence of the translation is that two types of formulas are excluded from the set of cut-formulas:

1. The formulas that are proper axioms.

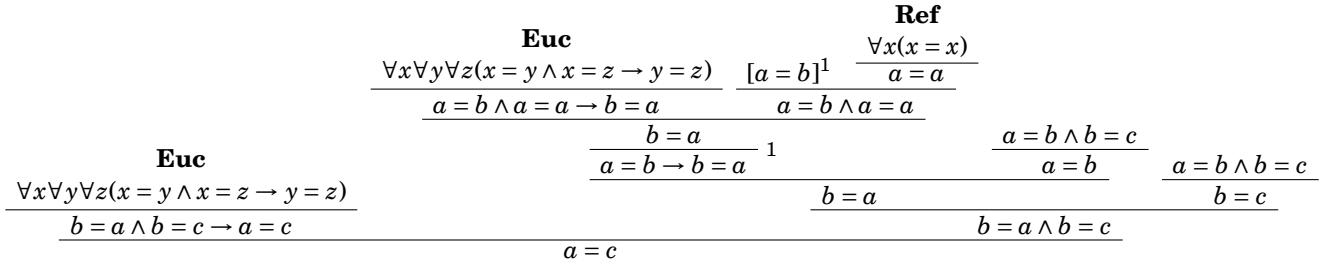
2. The principal formulas of logical axioms (Negri & von Plato 2001, p. 16), also called identity axioms (Girard et al. 1989, p. 30).

[3] For the notion of maximal formula see Dummett (1977, p. 152).

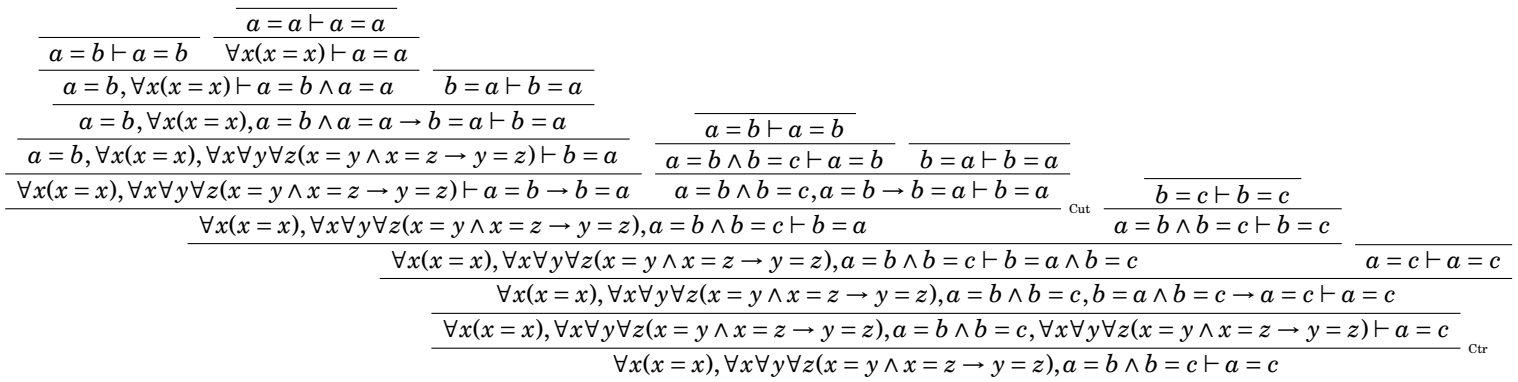
[4] Notice that the difference between proper axioms and assumptions is that the first can never be discharged, while the second are in principle always dischargeable (even if *de facto* they are not). At the level of proofs-objects – i.e. at the level of those objects used for codifying derivational steps, as λ-terms are (see Sundholm 1998, pp. 196-197) – the difference is that proper axioms correspond to proof-term constants while assumptions to proof-term variables. Roughly speaking, the idea is that proper axioms are sentences considered as already been proved, so that a justification can always be found for them (see Heyting 1962, p. 239). Assumptions, on the other hand, are place holders for proofs, in the sense that they wait to be replaced by a proof that neither we actually posses nor we know if it can ever be constructed.

[5] The translation works for systems of minimal, intuitionistic and classical logic. It is worth noting that Prawitz treats sequents as composed by sets of formulas. However, his translation can be adapted to the case of sequents considered as composed by multisets. In this case, the sequent calculus systems towards which the translation is directed are either **G1[mic]** or **G2[mic]** (cf. Troelstra & Schwichtenberg 2000, pp. 61-66 for a presentation of these systems).

[6] It is worth noting that differently from it, in Gentzen's translation (Gentzen 1934-35, §4) also normal proofs are translated into proofs with non-atomic cuts, because elimination rules are translated making appeal to a cut.



(a) Natural Deduction Proof



(b) Sequent Calculus Proof

Figure 1: Example of translation from natural deduction to sequent calculus

And because a fundamental property of any "good" sequent calculus system is the possibility of "atomizing" the principal formulas of identity axioms (Wansing 2000, pp. 10-11),[7] we can replace 2. by

2*. All the atomic formulas appearing in the logical (i.e. identity) axioms.

This means that from the computational point of view, proper axioms and (atomic) identity axioms are identified: both of them have no role in the execution of a program.[8] They have no genuine computational content, as they are just the external borders of proofs. In other terms, we can say that the *dynamics* of proofs, when it is conceived as cut-elimination, does not give any interesting information about the role played by both proper and identity axioms in formal proofs. But, what about if we look at other dynamical aspects of proofs, like proof-search procedures, i.e. bottom-up proof's reconstructions? Are these procedures more informative about the proof-theoretical role of axioms?

First, it should be noticed that by shifting the attention from the cut-elimination procedures – seen as a program executions – to proof-search procedures entails also a shift of logical frameworks, from intuitionistic[/constructive] logics to classical logic. The reason is that, with respect to proof-search, classical systems are much more suitable than intuitionistic ones, because of the invertibility of all their logical rules (Troelstra & Schwichtenberg 2000, p. 79). Let us then concentrate on classical sequent calculus and consider a provable sequent of the form $\Gamma, A \vdash \Delta$, where A is a proper axiom or an instance of a proper axiom scheme. If we dispose of an algorithmic procedure allowing to reconstruct the proof of the sequent, for example by working with a system of classical logic like **G3c**, the best we can do is to decompose A into atomic sentences belonging to some initial identity axioms.[9] Again, we should conclude that proper axioms have no particular role in proofs: they are not different from other context formulas used in purely logical proofs, and everything can be eventually reduced to logical combinations of identity axioms. In order to prevent from this transformation of proofs containing proper axioms into purely logical proofs, we have to block the logical decomposition of the axiom A. A very rough idea would be to apply a proof-search procedure on $\Gamma, A \vdash \Delta$ within a system deprived of left-rules. In fact, this is completely equivalent to looking for proofs in a right-handed system with an additional initial sequent $\vdash A$. However, in a such a system, every proof using $\vdash A$ actually uses cuts that are not eliminable (Girard 1987a, p. 125,

[7] It is worth noting that as cut-elimination corresponds to program execution – i.e. β-reduction –, also this property of identity axiom atomization possess a well defined computational content. Namely, it corresponds to what is called η-expansion, a property that guarantees the possibility of working in an extensional setting even in the case of programs, which by definition are instead intensional objects (cf. Hindley & Seldin 2008, pp. 76-77).

[8] The λ-term associated to the previous natural deduction proof is $(\mathtt{t})\langle(\lambda x(\mathtt{t})\langle x,\mathtt{r}\rangle)\pi_1(z),\pi_2(z)\rangle$, where $\mathtt{r}$ and $\mathtt{t}$ are two proof-constants respectively associated to the reflexivity and the euclidean axioms. Reducing the β-*redex* contained in this λ-term – which corresponds to the detour of the proof that this λ-term codifies – we get $(\mathtt{t})\langle(\mathtt{t})\langle\pi_1(z),\mathtt{r}\rangle,\pi_2(z)\rangle$. It is not difficult to see that the constants $\mathtt{t}$ and $\mathtt{r}$, as well as the variable z, are not involved in the process of reduction. This means that proof-constants have no interaction with the other proof-constructors and that we cannot assign to them any computational content. Proof-objects in this case have only the role of codifying the structure of the proofs to which they are associated with, but they cannot be interpreted as programs.

[9] In terms of proof-objects this point becomes absolutely clear. While in natural deduction the proof-objects associated to proper axioms are constants, in sequent calculus they are complex λ-terms not containing any constant, because proper axioms themselves have been constructed in the context of derivations, i.e. in the antecedent.

Troelstra & Schwichtenberg 2000, p. 127). But, in general, cuts are an obstacle to the root-first reconstruction of proof, because if we want to determine whether a sequent $\Gamma \vdash \Delta$ is derivable, by using a cut rule we should check the derivability of the two sequents $\Gamma \vdash C$ and $C, \Gamma \vdash \Delta$ for any arbitrary formula C, and the immediate consequence is the lack of any bound on the proof-search. Hence, a proof-search procedure allowing the recognition of all theorems of the theory T containing the axiom A is not always terminating. However, we could still operate a proof search on a given theorem B, belonging to a certain theory T, without requiring to close the derivation, that is to say without necessarily using the axioms of T as initial sequent of the form $\vdash A$. More precisely, suppose B to be a formula for which there are no positive occurrences of existential formulas and no negative occurrences of universal formulas.[10] When we work in **G3c**, a sequent $\vdash B$ having empty antecedent can always be univocally decomposed in a set of *basic sequents* of the form (Negri & von Plato 2001, p. 50)

$$\bot, \ldots, \bot, P_1, \ldots, P_m \vdash Q_1, \ldots, Q_n, \bot, \ldots, \bot \tag{1}$$

where P_i and Q_j are atoms.

If B is a non-logical theorem, or even an axiom, then its decomposition leads to a (possibly infinite) set of «basic mathematical sequents» (Gentzen 1938, p. 257) containing at least one sequent of the form (Negri & von Plato 2001, p. 51)

$$P_1, \ldots, P_m \vdash Q_1, \ldots, Q_n, \bot, \ldots, \bot \tag{2}$$

where $P_i \not\equiv Q_j$ for every i and j.

It is worth noting that atomic identity axioms are just a particular case of (1), namely when there are no $\bot$ and $P_i \equiv Q_j$ for some i and j. This remark suggests the possibility of defining a unique way to deal both with proper axioms and identity axioms. In the next section we propose a solution going in this direction by introducing a *generalized* axiom rule inspired by Girard (2001).

3 From proofs to models

As already anticipated, in this section the attention will be focused on classical logic. This choice is not simply due to practical – if not even opportunistic – reasons related to the aforementioned efficacy of classical systems over intuitionistic ones with respect to proof-search problems. There is in fact a deeper and more conceptual reason connected to what we said in §1. As we mentioned before, our aim is to stick as much as possible to a non-revisionist position with respect to the architecture of mathematical theories, and one of the characteristic features of the standard view is precisely that the logic subtending mathematical theories is classical logic, and not intuitionistic logic.

3.1 Schütte's completeness proof revisited

We introduce a system useful in order to study from the unifying perspective of Schütte's completeness proof (see Schütte 1956) both the syntactical and the semantical role played both by identity and proper axioms. In fact, properly speaking, our system should be considered more as a general framework for carrying

[10] For the standard definition of positive and negative occurrence of a formula see Troelstra & Schwichtenberg (2000, p. 6).

out an abstract and formal study of axioms rather than as a genuine deductive system. As we will see, the reason is that in order to have a sufficient expressive power for speaking of every possible axioms we are obliged to flirt with inconsistency. Nonetheless the proper logical part of the system can be singled out through the definition of a kind of correctness criterion.

We start by presenting the propositional case, namely the system $\text{pLK}^{\Pisymbol{pzdr}{120}}_R$. We will then move to the more interesting case of first order logic. However, in order to gather the main characteristic features of the systems and to understand how it works it is sufficient to look at the propositional part. Who is not interested in a finer analysis of the system can then skip the predicative part.

Let $\mathscr{A}$ be a set of atomic formulas.

Definition 1 (Formulas and Sequents)**.** The set of formulas is inductively defined by the following grammar

$$F := P, Q \mid \neg P \mid F \vee F \mid F \wedge F \qquad (P, Q \in \mathscr{A})$$

A sequent $\Gamma \vdash \Delta$ is an ordered pair of multisets Γ, Δ of formulas.

Definition 2. The system $pLK^{\Pisymbol{pzdr}{120}}_R$ is defined by the rules of figure 2.

$$\frac{}{\vdash \Gamma}\ \Pisymbol{pzdr}{120}^{At} \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \vee B}\ \vee \qquad \frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \wedge B}\ \wedge$$

Figure 2: $\text{pLK}^{\Pisymbol{pzdr}{120}}_R$ rules

The only proviso on the application of $\Pisymbol{pzdr}{120}^{At}$, the generalized axiom rule, is that the formulas appearing in the sequent $\vdash \Gamma$ (if there are any) have to be atomic formulas or negations of atomic ones.

Proposition 3 (Invertibility)**.** *The rules* $\vee$ *and* $\wedge$ *are invertible.*

Proof. By induction on the number of connectives in the sequent. Both proofs (for $\vee$ and for $\wedge$) being similar, we only show the invertibility of the $\vee$. The base case for the induction is a sequent containing only one connective, i.e. a $\Pisymbol{pzdr}{120}$ rule followed by a $\vee$-rule. In this case the $\Pisymbol{pzdr}{120}^{At}$ rule itself gives a derivation of the premiss of the $\vee$-rule. Let us now assume that this is true for any sequent containing at most n connectives, and let us take a derivable sequent $\vdash \Gamma, A \vee B$, containing $n+1$ connectives, and let π be one of its derivations. If π ends with a $\vee$-rule, the subderivation obtained dropping the last rule gives a derivation of the premiss. If π does not end with a $\vee$ rule, then it must end with a $\wedge$ rule:

$$\frac{\begin{array}{c}\pi_1 \\ \vdots \\ \vdash \Delta, A \vee B, C\end{array} \qquad \begin{array}{c}\pi_2 \\ \vdots \\ \vdash \Delta, A \vee B, D\end{array}}{\vdash \Delta, A \vee B, C \wedge D}\ \wedge$$

Applying the induction hypothesis on the premisses we get two derivations π'_1 and π'_2 of $\vdash \Delta, A, B, C$ and $\vdash \Delta, A, B, D$ respectively; thus the desired derivation is:

$$\dfrac{\begin{array}{cc}\pi'_1 & \pi'_2\\ \vdots & \vdots\\ \vdash \Delta, A, B, C & \vdash \Delta, A, B, D\end{array}}{\vdash \Delta, A, B, C \wedge D}\ \wedge$$

□

Proposition 4. *Every sequent can be derived in* $\mathrm{pLK}^{\maltese}_{R}$.

Proof. By trivial induction on the number of connectives in the sequent. □

Definition 5. Given a proof π in $\mathrm{pLK}^{\maltese}_{R}$, $\mathfrak{L}(\pi)$ is the multiset of sequents introduced by the $\maltese$ rules in π. $\mathfrak{L}(\pi)$ is called the *set of leaves* of π.

Lemma 6 (Adequation/Completeness)**.** *Let* π *and* π' *be derivations of* $\vdash \Gamma$ *in* $\mathrm{pLK}^{\maltese}_{R}$*, then* $\mathfrak{L}(\pi) = \mathfrak{L}(\pi')$.

Proof. By induction on the number of connectives in $\vdash \Gamma$. The base case is obvious. Assume the lemma to be valid when the number of connectives in Γ is at most n. What we show is that the derivations of $\vdash \Gamma, F, G$ terminating with a rule where F is principal have the same set of leaves as the derivations terminating with a rule where G is principal. We only show how the proof is done in the case $F = A \wedge B$ and $G = C \wedge D$, which is the more complicated case. By the invertibility of the $\wedge$-rule we have the two derivations:

$$\dfrac{\dfrac{\dfrac{\begin{array}{cc}\pi_1 & \pi_2\\ \vdots & \vdots\\ \vdash \Gamma, A, C & \vdash \Gamma, A, D\end{array}}{\vdash \Gamma, A, C \wedge D}\ \wedge \qquad \dfrac{\begin{array}{cc}\pi_3 & \pi_4\\ \vdots & \vdots\\ \vdash \Gamma, B, C & \vdash \Gamma, B, D\end{array}}{\vdash \Gamma, B, C \wedge D}\ \wedge}{\vdash \Gamma, A \wedge B, C \wedge D}\ \wedge$$

$$\dfrac{\dfrac{\dfrac{\begin{array}{cc}\rho_1 & \rho_2\\ \vdots & \vdots\\ \vdash \Gamma, A, C & \vdash \Gamma, B, C\end{array}}{\vdash \Gamma, A \wedge B, C}\ \wedge \qquad \dfrac{\begin{array}{cc}\rho_3 & \rho_4\\ \vdots & \vdots\\ \vdash \Gamma, A, D & \vdash \Gamma, B, D\end{array}}{\vdash \Gamma, A \wedge B, D}\ \wedge}{\vdash \Gamma, A \wedge B, C \wedge D}\ \wedge$$

Using the induction hypothesis, we have that $\mathfrak{L}(\pi_k) = \mathfrak{L}(\rho_k)$ for all k in $\{1,2,3,4\}$. Using these equalities and the induction hypothesis once again, we obtain that:

- any derivation π of $\vdash \Gamma, A, C \wedge D$ satisfies $\mathfrak{L}(\pi) = \mathfrak{L}(\pi_1) + \mathfrak{L}(\pi_2)$;
- any derivation π of $\vdash \Gamma, B, C \wedge D$ satisfies $\mathfrak{L}(\pi) = \mathfrak{L}(\pi_3) + \mathfrak{L}(\pi_4)$;
- any derivation π of $\vdash \Gamma, A \wedge B, C$ satisfies $\mathfrak{L}(\pi) = \mathfrak{L}(\pi_1) + \mathfrak{L}(\pi_3)$;
- any derivation π of $\vdash \Gamma, A \wedge B, D$ satisfies $\mathfrak{L}(\pi) = \mathfrak{L}(\pi_2) + \mathfrak{L}(\pi_4)$;

We therefore have shown that if π is any derivation of $\vdash \Gamma, A \wedge B, C \wedge D$ terminating with a $\wedge$-rule on $A \wedge B$ and, if ρ is any derivation of the same sequent terminating with a $\wedge$-rule on $C \wedge D$, they have the same set of leaves, namely:

$$\mathfrak{L}(\pi) = \sum_{i=1,\ldots,4} \mathfrak{L}(\pi_i) = \mathfrak{L}(\rho)$$

The other cases are done similarly. □

Remark 7. By Proposition 4 and the previous lemma it makes sense to talk of $\mathfrak{L}(\vdash \Gamma)$, for any sequent $\vdash \Gamma$.

Definition 8. A sequent $\vdash \Gamma$ is *correct* when it is atomic and there exists an atom P, such that P and $\neg P$ are both in $\vdash \Gamma$. By extension, a $\maltese^{At}$ rule is correct when the sequent it introduces is correct.

An incorrect sequent is an atomic sequent that is not correct.

Definition 9. The system pLK_R is obtained by replacing the $\maltese^{At}$ rule with a rule that introduces only correct sequents. This rule will be called logical axiom rule and noted in the following manner

$$\frac{}{\vdash \Gamma, P, \neg P}\ \mathrm{ax}$$

Proposition 10. *A sequent $\vdash \Gamma$ is derivable in pLK_R if and only if $\mathfrak{L}(\vdash \Gamma)$ contains only correct sequents.*

Proof. Suppose π is a derivation of $\vdash \Gamma$ in pLK_R, then by replacing every axiom rule by a $\maltese$ rule we obtain a derivation π' of $\vdash \Gamma$ in $\mathrm{pLK}_R^{\maltese}$. Every $\maltese^{At}$ rule in π' is correct, since the sequent was introduced by an axiom rule in pLK_R.

Conversely, if π' is a derivation of $\vdash \Gamma$ in $\mathrm{pLK}_R^{\maltese}$ such that $\mathfrak{L}(\pi')$ contains only correct sequent, then each sequent in $\mathfrak{L}(\pi')$ can be derived from an axiom rule in pLK_R. Therefore we obtain a derivation π of $\vdash \Gamma$ in pLK_R by replacing every $\maltese^{At}$ rule in π' by an axiom rule. □

Definition 11. Let $\delta : Prop \to \{0,1\}$ be a valuation.

- $\delta \vDash \Gamma$ if and only if there exists at least one $A \in \Gamma$ such that $\delta(A) = 1$
- $\vDash \Gamma$ if and only if for all δ, $\delta \vDash \Gamma$.

Lemma 12. *For any valuation δ, $\delta \vDash \Gamma$ if and only if for all $\vdash \Delta$ in $\mathfrak{L}(\vdash \Gamma)$, $\delta \vDash \Delta$.*

Proof. Notice that, by definition of satisfiability of a sequent (and associativity of $\vee$), $\delta \vDash \Delta, A, B$ if and only if $\delta \vDash \Delta, A \vee B$.

In the case of the $\wedge$-rule, assume first that $\delta \vDash \Delta, A$ and $\delta \vDash \Delta, B$. Either there is satisfiable formula in Δ, either both A and B are satisfiable, therefore $\delta \vDash \Delta, A \wedge B$. Conversely, assume that $\delta \nvDash \Delta, A$, then all formulas in Δ are unsatisfiable and A is not satisfiable, therefore $A \wedge B$ is not satisfiable. We conclude that $\delta \nvDash \Delta, A \wedge B$. The lemma is then proved by simple induction. □

Proposition 13. *$\vDash \Gamma$ if and only if all sequents in $\mathfrak{L}(\vdash \Gamma)$ are correct.*

Proof. Suppose $\mathfrak{L}(\vdash \Gamma)$ contains only correct sequents, then for any valuation δ and sequents $\vdash \Delta$ in $\mathfrak{L}(\vdash \Gamma)$, $\delta \vDash \Delta$ from the definition of correct sequent. Then, by Lemma 12, $\delta \vDash \Gamma$.

Conversely, let us assume that $\mathfrak{L}(\vdash \Gamma)$ contains at least an incorrect sequent $\vdash P_1, \ldots, P_n, \neg Q_1, \ldots \neg Q_p$, such that for all integer $i \in [1,n]$ and $j \in [1,p]$, $P_i \neq Q_j$. We can now take a valuation δ satisfying $\delta(P_i) = 0$ and $\delta(Q_j) = 1$. Then $\delta \nvDash P_1, \ldots, P_n, \neg Q_1, \ldots \neg Q_p$ and by Lemma 12 this means that $\delta \nvDash \Gamma$. □

3.2 Embedding semantics in the syntax

We adapt the preceding proof to the first-order case. We must take particular care of the ✠ rule and of the definition of correctness. As it concerns the ✠ rule notice that differently from the propositional case, here we are even more liberal: no conditions are imposed on the application of the rule; it can be used at every point of the derivation.

Definition 14. Let $\mathrm{LK}_R^{✠}$ be the system defined by the rules of figure 3.

$$\frac{}{\vdash \Gamma}\ ✠$$

$$\frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \vee B}\ \vee \qquad \frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \wedge B}\ \wedge$$

$$\frac{\vdash \Gamma, A[y/x]}{\vdash \Gamma, \forall x A(x)}\ \forall \quad (y \text{ fresh}) \qquad \frac{\vdash \Gamma, A(t), \exists x A(x)}{\vdash \Gamma, \exists x A(x)}\ \exists$$

Figure 3: The rules of $\mathrm{LK}_R^{✠}$ sequent calculus

Definition 15. A derivation of $\mathrm{LK}_R^{✠}$ is a finite tree obtained from the rules of figure 3 such that all leaves terminate with a ✠ rule.

Proposition 16. *Every sequent $\vdash \Gamma$ is derivable in $LK_R^{✠}$.*

Proof. Take $\vdash \Gamma$ and close the derivation by an instance of ✠ rule. □

Definition 17. The ✠ rule that introduces the sequent $\vdash \Gamma$ is correct if there exists a formula A such that both A and $\neg A$ are in Γ.

A derivation is correct if all its ✠ rules are correct.

Definition 18. A ✠ rule introducing a sequent $\vdash \Gamma$ is *admissible* if either it is correct or there exists a correct derivation of $\vdash \Gamma$ in $\mathrm{LK}_R^{✠}$.

Lemma 19. *If there exists a derivation π of $\vdash \Gamma$ in $\mathrm{LK}_R^{✠}$ such that $\mathfrak{L}(\pi)$ contains only admissible ✠ rules, then π can be extended to a derivation π' of $\vdash \Gamma$ such that $\mathfrak{L}(\pi')$ contains only correct ✠ rules.*

Proof. Let π be such a derivation of the sequent $\vdash \Gamma$. Let $\mathfrak{L}(\pi)$ be the set of sequents introduced by ✠ rules in $\vdash \Gamma$, $\mathfrak{L}_c(\pi)$ the subset of $\mathfrak{L}(\pi)$ containing the sequents introduced by correct ✠ rules, and $\mathfrak{L}_a(\pi) = \mathfrak{L}(\pi) - \mathfrak{L}_c(\pi)$. By assumption, the sequents in $\mathfrak{L}_a(\pi)$ are introduced by admissible ✠ rules that are not correct. Hence there exists correct derivations π_i of $\vdash \Gamma_i$. Then, replacing the ✠ rules introducing the sequents $\vdash \Gamma_i$ in π by the derivations π_i, we obtain a derivation π' of $\vdash \Gamma$ extending π and containing only correct ✠ rules. □

Definition 20. The system LK_R is obtained by replacing the ✠ rule with a rule that introduces only correct sequents. This rule will be called logical axiom rule and noted in the following manner

$$\frac{}{\vdash \Gamma, A, \neg A}\ \mathrm{ax}$$

Theorem 21. *The sequent $\vdash \Gamma$ is derivable in LK_R if and only if there exists a derivation π of $\vdash \Gamma$ in $\mathrm{LK}_R^{✠}$ and $\mathfrak{L}(\pi)$ contains only admissible ✠ rule.*

Proof. Suppose we have a derivation of π in LK_R of a sequent $\vdash \Gamma$. Then, replacing every axiom rule by a ✠ yields a derivation π' of $\vdash \Gamma$ in $\mathrm{LK}_R^{✠}$. Moreover, the ✠ rules are all correct (since they were axiom rules in LK_R), hence admissible.

Conversely, suppose we have a derivation π' of a sequent $\vdash \Gamma$ in $\mathrm{LK}_R^{✠}$ that contains only admissible rules. Then, by lemma 19 we can find a derivation π'' extending π' such that π'' contains only correct ✠ rules. Then, we can replace these ✠ rules by axiom rules to get a derivation π of $\vdash \Gamma$ in LK_R . □

Definition 22. A ✠ rule is *simple* if the sequent $\vdash \Gamma$ it introduces contains only atoms, negations of atoms and existential formulas.

A *simple derivation* is a derivation in which all ✠ rules are simple.

Lemma 23. *Let π be a derivation in $LK_R^{✠}$. There exists a simple extension of π.*

Proof. Suppose π contains at least one ✠ rule introducing a sequent $\vdash \Gamma$ containing a formula B that is not an atom, a negation of an atom or an existential formula. Then the principal connective in B is either a $\wedge$, a $\vee$ or a $\forall$. Replacing the ✠ rule introducing $\vdash \Gamma$ by the rule introducing the principal connective and closing the derivation we obtain by ✠ rules then gives us a new derivation π_1 of π. After a finite number of iterations of this process, we obtain the wanted extension. □

Definition 24. Let $B = \exists x A$ be an existential formula. We define the *instances* of B to be the formulas $A[t/x]$ where t is a term.

More generally, if A is a formula and C a subformula of A, the set of *instances* of C is the set of formulas $C[t_1/x_1,\ldots,t_n/x_n]$ where $t_1,\ldots,t_n$ are terms and $x_1,\ldots,x_n$ are the bounded variables of C.

Lemma 25. *Let π be the following derivation:*

$$\frac{}{\vdash \Gamma}\ ✠$$

If π is non admissible, then we can find a sequence of extensions of π containing all the instances of the subformulas of Γ.

Proof. Suppose now that the sequent $\vdash \Gamma$, $\Gamma = A_1,\ldots,A_m$, introduced by the non-admissible ✠ rule contains at least one quantifier. We fix an enumeration of the terms $t_1,\ldots,t_n,\ldots$ of the language and we will define an iterative process indexed by couples (s,k_s) where s is a finite sequence of integers (the first step will be indexed by the null sequence of length m which will be written $(0)_m$) of length p_s and k_s is an integer in $[1,\ldots,p_s]$. The process we describe consists in extending the derivation by applying $\exists$ rules in a way that insures us that for all existential formula $\exists x A(x)$ and term t_i, there exists a step where the $\exists$ rule is used on the formula $A[t_i/x]$. To insure all terms appear at some point in the process we will use the enumeration but we need to keep track of the last term used for each existential formula. Moreover, applying a $\exists$ rule on a formula containing two existential connectives will produce new existential formulas on which we must apply the same procedure. The sequence will therefore keep track, for each existential formula, of the last term we used. Its length may vary, but due to our

choice of existential rule it can only expand. The integer, on the other hand, will keep track of the last existential formula we decomposed, so that we can ensure that all formulas are taken into account.

First, let us write $A_1,\dots,A_{p_{(0)_m}}$ the formulas in Γ that contain quantifiers. By lemma 23 we can suppose, without loss of generality, that the ✠ rules in π are simple. We will denote π by $\pi_{(0)_m}$, i.e. π will be the initial step of the process. The integer $k_{(0)_m}$ is defined to be 1, so we consider the derivation π_{A_1,t_1} obtained from π by replacing the ✠ rule introducing $\Gamma = \Delta, A_1,\dots,A_p$ by the derivation consisting of a ✠ rule introducing $\Delta, A'_1[t_1/x], A_2,\dots,A_{p_{(0)_m}}$ followed by an existential rule introducing A_1. It follows from Lemma 23 that this derivation can be extended to a simple derivation $\bar{\pi}_{A_1,t_1}$. Then, by the non-admissibility of the ✠ rule, this derivation contains at least one non-admissible ✠ rule introducing a sequent $\vdash \Gamma'$. Amongst the formulas of Γ are the all the formulas A_i for $1 \leqslant i \leqslant p$, but Γ' may contain more existential formulas. We thus denote by $A_1,\dots,A_p$ the existential formulas of Γ'. We write $(0)^+_m = (1,0,\dots,0)$ the sequence of length p: we thus obtained an extension $\pi_{(0)^+_m} = \bar{\pi}_{A_1,t_1}$ containing a non-admissible ✠ rule introducing a sequent $\Gamma_{(0)^+_m} = \bar{\Gamma}$. Defining $k_{(0)^+_m} = 2$, we arrived at the next step, indexed by $((0)^+_m, k_{(0)^+_m})$ and we can then iterate the process.

More generally, suppose we are at step (s, k_s) with $s = (s(0),\dots,s(p))$: we have a simple derivation π_s with a non-admissible ✠ rule introducing a sequent $\Gamma_s = \Delta_s, A_1,\dots,A_{p_s}$ (Δ_s contains only atoms and negations of atoms). We obtain a derivation $\pi_{A_{k_s},t_{s(k_s)+1}}$ by replacing the ✠ rule introducing Γ_s with the derivation:

$$\dfrac{\dfrac{}{\vdash \Delta_s, A_1,\dots,A_{p_s}, A'_{k_s}[t_{s(k_s)+1}]}\,✠}{\vdash \Gamma_s}\,\exists$$

This derivation $\pi_{A_{k_s},t_{s(k_s)+1}}$ can then be extended by Lemma 23 to a simple derivation $\bar{\pi}_{A_{k_s},t_{s(k_s)+1}}$ which contains a non-admissible ✠ rule. The sequent Γ' introduced by this rule contains all the formulas $A_1,\dots,A_{p_s}$ and may contain additional existential formulas $A_{p_s+1},\dots,A_n$. Let s^+ to be the sequence of length n defined by $(s(0),\dots,s(k_s-1), s(k_s)+1, s(k_s+1),\dots,s(p_s),0,\dots,0)$, and:

$$k_{s^+} = \begin{cases} k_s + 1 & \text{if } k_s + 1 \leqslant n \\ 1 & \text{otherwise} \end{cases}$$

Let us write $n = p_{s^+}$. We thus obtained the next step in the process, indexed by (s^+, k^+_s): a simple derivation $\pi_{s^+} = \bar{\pi}_{A_{k_s},t_{s(k_s)+1}}$ with a non-admissible ✠ rule introducing a sequent $\Gamma_{s^+} = \Gamma' = \Delta_{s^+}, A_1,\dots,A_{p_{s^+}}$.

We claim that for all couple (i,j) of natural numbers (different from 0), there is a step s in the process such that $s(i) = j$. We will write $\mathrm{len}(s)$ the length of a sequence s. Notice the formulas $A_{p_s+1},\dots,A_{p_{s^+}}$ are instantiations of an existential formulas A_i for $1 \leqslant i \leqslant p_s$ and therefore the number of existential connectives in a B_j is strictly less than the number of existential connectives in the corresponding A_i. We will show that the value of k_s returns to 1 in a finite number of steps using this remark. We will denote the number of existential connectives in a formula A by $♮(A)$. Suppose that we are at a given step s such that $k_s = 1$ and write $o_s = (\max_{k_s < i \leqslant p_s} ♮(A_i), p_s - k_s)$. This couple somehow measures the number of steps one has to make before k_s returns to 1. Then, after $p_s - k_s$ steps in the process – let us write the resulting step as s^1, we have $k_{s^1} = p_s$ and $p_{s^1} - p_s$ new formulas, each one such that $♮(A) < \max_{k_s < i \leqslant p_s} ♮(A_i)$. Therefore, the couple

$o_{s^1} = (\max_{k_{s^1} < i \leqslant p_{s^1}} \natural(A_i), p_{s^1} - p_s)$. Since $\max_{k_{s^1} < i \leqslant p_{s^1}} \natural(A_i) < \max_{k_s < i \leqslant p_s} \natural(A_i)$, we have $o_{s^1} < o_s$ in the lexicographical order and this is enough to show the claim. □

Theorem 26. *Let π be a derivation in $LK_R^{\maltese}$ of a sequent $\vdash \Gamma$ containing a non-admissible $\maltese$ rule. Then there exists a model $\mathscr{M}$ such that $\mathscr{M} \not\models \Gamma$.*

Proof. If the sequent $\vdash \Gamma$ introduced by the non-admissible $\maltese$ rule does not contain any quantifiers, then the proof reduces to the proof of Proposition 13. Indeed, the derivation of $\vdash \Gamma$ in $pLK_R^{\maltese}$ is incorrect (if it were correct, it would contradict the assumption since any correct derivation in $pLK_R^{\maltese}$ is a correct derivation in $\mathrm{LK}_R^{\maltese}$), hence we can find a model $\mathscr{M}$ such that $\mathscr{M} \not\models \Gamma$.

If $\vdash \Gamma$ contains existential formulas, we use Lemma 25 to obtain a sequence $(\pi_i)_{i \in \mathbf{N}}$ of extensions. From this sequence of extensions, one can obtain a sequence of sequents $\vdash \Gamma_i$ where for each i, there exists N such that $\vdash \Gamma_{i+1}$ is the premise of a rule whose conclusion is $\vdash \Gamma_i$ in all derivations π_j with $j \geqslant N$. Moreover, this sequence can be chosen so as to contain all instances of the subformulas of Γ. We now define a model whose base set is the set of terms. The interpretations of function symbols and constants are straightforward. The only thing left to define is the interpretation of predicates: if P is a n-ary predicate symbol, then $(t_1, \ldots, t_n)$ is in the interpretation of P if and only if $\forall i \geqslant 0$, $Pt_1 \ldots t_n \notin \Gamma_i$.

We can now check that $\mathscr{M} \not\models \Gamma$. We chose A a formula in Γ and prove by induction on the size of the formula A that $\mathscr{M} \not\models A$:

- if A is an atomic formula, then $\mathscr{M} \not\models A$ by definition of the model;
- if $A = B \wedge C$, then there exists a sequent $\vdash \Gamma_i$ such that either $B \in \Gamma_i$ or $C \in \Gamma_i$. We suppose $B \in \Gamma_i$ without loss of generality. Then, by the induction hypothesis, we have $\mathscr{M} \not\models B$, hence $\mathscr{M} \not\models A$;
- if $A = B \vee C$, then there exists a sequent $\vdash \Gamma_i$ containing both B and C. By induction, these two formulas are not satisfied in the model $\mathscr{M}$, hence $\mathscr{M} \not\models A$;
- if $A = \forall x B(x)$, then there is a sequent $\vdash \Gamma_i$ containing $B[y/x]$. By induction, $\mathscr{M} \not\models B[y/x]$, hence $\mathscr{M} \not\models A$;
- if $A = \exists x B(x)$, then for every term t there exists a sequent $\vdash \Gamma_i$ such that $B[t/x] \in \Gamma_i$. By the induction hypothesis, $\mathscr{M} \not\models B[t/x]$. This being true for all term t, we conclude that $\mathscr{M} \not\models A$.

This concludes the proof: since all formula $A \in \Gamma$ is such that $\mathscr{M} \not\models A$, we have that $\mathscr{M} \not\models \Gamma$. □

Theorem 27. *If π is a derivation of $\vdash \Gamma$ in $LK_R^{\maltese}$ containing only admissible $\maltese$ rules, then for all model $\mathscr{M}$, $\mathscr{M} \models \Gamma$.*

Corollary 28. *Let π and π' be derivations in $LK_R^{\maltese}$ of the same sequent $\vdash \Gamma$. Then π contains only admissible $\maltese$ rules if and only if π' contains only admissible $\maltese$ rules.*

3.3 Axioms and models

Differently from what happens with standard Schütte's completeness proof for classical logic, in the revisited proof we have just proposed when $\vdash \Gamma$ is derived with non-admissible instances of ✠ rule, we cannot always conclude that $\bigvee\Gamma$ is an antilogy, that is a sentence which is false in every possible model.[11] In fact, $\bigvee\Gamma$ could be valid in some particular theories, namely in every theory T the models of which render true every non-admissible instances of ✠ rule used in the derivation of $\bigvee\Gamma$. The derivation of $\bigvee\Gamma$ is then a source of information about the axioms and theorems of T. In particular, the non correct instances of ✠ rule correspond to a set of sequents $\mathscr{S}_1,\ldots,\mathscr{S}_n$ such that

i. either each $\mathscr{S}_i$ is provable in every axiomatic system containing Γ,

ii. or $\vdash \Gamma$ is provable in every axiomatic system containing $\mathscr{S}_1,\ldots,\mathscr{S}_n$.

To sum up, the proof search in the system $\mathrm{LK}^{✠}_{R}$ does not always represents a method for invalidating non-logical sentences, but it can also be used to make explicit the set of conditions under which a non-tautology – i.e. a sentence which is a theorem of a particular theory T[12] – is valid. In particular, the role played by the non correct instances of ✠ rule is to identify that particular class of models validating the non-tautology into question. Differently from what happens with tautologies or antilogies, where either the whole class or the empty class of models is considered, in the case of non-tautologies the instances of the ✠ rule select a very specific and non-trivial class of models. It seems then that the proof search dynamics leads to corroborate the standard view presented in §1 according to which the role of proper axioms is to identify classes of relational structures. However, here the conceptual order is inverted: structures are no more primitive entities, which are later syntactically fixed by axioms, but they become themselves entities that are generated from the syntactical features of the proof search dynamics.

Nevertheless, these structures are not yet homogeneous with syntactical entities, since they are still set-theoretical entities. In what follows, we will try to present a general framework allowing to treat proofs and models – or better, countermodels – from a homogeneous point of view. In order to do that, we need to liberalize the usual notion of syntax. To clarify this statement, let us start with the problem of distinguishing between antilogies and non-tautologies.

4 Distinction between non-tautologies and antilogies

Despite of what we said in the previous section, the framework of $\mathrm{LK}^{✠}_{R}$ does not yet allow by itself to properly distinguish between sequents that are non-tautologies and antilogies. Let us consider the two following derivations:

[11] In other words, an antilogy is the negation of a tautology.

[12] In the literature, this kind of formulas are usually called *neutral formulas*. However, we prefer to avoid this terminology because from our perspective, even if these formulas are neutral form a logical point of view – as they are neither tautology nor antilogy –, they are not neutral from the point of view of a particular theory T. And since in this paper we are interested in specific mathematical theories, then calling these formulas neutral could be a source of confusion.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \neg A, B}\,\text{✠}}{\vdash \neg A \vee B}\,\vee \qquad \dfrac{}{\vdash A}\,\text{✠}}{\vdash (\neg A \vee B) \wedge A}\,\wedge \qquad \dfrac{\dfrac{}{\vdash \neg B}\,\text{✠} \qquad \dfrac{}{\vdash \neg C}\,\text{✠}}{\vdash \neg B \wedge \neg C}\,\wedge}{\vdash ((\neg A \vee B) \wedge A) \wedge (\neg B \wedge \neg C)}\,\wedge$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, \neg B}\,\text{✠}}{\vdash A \vee \neg B}\,\vee \qquad \dfrac{}{\vdash A}\,\text{✠}}{\vdash (A \vee \neg B) \wedge A}\,\wedge \qquad \dfrac{\dfrac{}{\vdash \neg B}\,\text{✠} \qquad \dfrac{}{\vdash \neg C}\,\text{✠}}{\vdash \neg B \wedge \neg C}\,\wedge}{\vdash ((A \vee \neg B) \wedge A) \wedge (\neg B \wedge \neg C)}\,\wedge$$

The simple observation that the two proofs appeal to non correct instances of the ✠ rule does not tell us anything about the fact that the conclusion sequent is an antilogy or a non-tautology. The only way to distinguish between antilogies and non-tautologies is to check if there exists a valuation rendering all ✠ rules true. For instance, in the previous example, such a valuation exists in the case of the second proof (making A true, and B and C false), while there exists none in the case of the first: we can thus conclude that the second proof is a non-tautology and the first one is an antilogy. The problem of this way of distinguishing between non-tautologies and antilogies is that it makes appeal to the inspection of all possible valuations of all the ✠ rules present in the proof. This method is then not exclusively based on proofs' inspection and, moreover, it is not really effective (especially when dealing with first order). What we want, instead, is to be able to judge if a sentence is a non-tautology or an antilogy effectively through a simple mechanical inspection of the proof. A possible way to do that is to analyze not only the proof of the sentence into question, but also the proof of its negation.

First, it is worth recalling that we are working in a framework where everything is derivable: no particular kinds of constraints have been indeed imposed on the application of the ✠^{At} rule. In particular, it could be applied also when $\Gamma = \emptyset$, and thus the empty sequent "$\vdash$" can be derived in $\text{pLK}^{\text{✠}}_{R}$. But the empty sequent represents in the object language the idea that an unspecified absurdity[13] – namely, the empty succedent – is deducible from any kind of hypothesis – namely, the empty antecedent (see Paoli 2002, p. 32). Deriving the empty sequent corresponds then to derive an absurdity as a theorem and thus to show that $\text{pLK}^{\text{✠}}_{R}$ is inconsistent. In order to prevent the system from being inconsistent an *ad hoc* solution is to impose a constraint on the application of the ✠^{At} rule, namely that $\Gamma \neq \emptyset$. In this way, even if any formula can still become a theorem, the system remains consistent; we would be then in a situation complementary to the one advocated by paraconsistent logics: $\text{pLK}^{\text{✠}}_{R}$ would be a trivial but consistent system. Actually, the only way to obtain an empty sequent would be to make appeal to the cut rule, allowing to get the empty sequent from the derivable sequents $\vdash A$ and $\vdash \neg A$, for a certain A. This would immediately allow a proof-theoretical characterization of absurdity as something for which we do not possess a canonical derivation, that is a derivation terminating with the rule corresponding to the principal connective of the conclusion-formula. This is exactly the explanation of absurdity given by verificationist accounts (see Sundholm 1983, p. 485; Martin-Löf 1996, p. 51). However, from our perspective we

[13] Absurdity is seen here in a Brouwerian perspective, that is as something that does not allow to continue a derivation anymore (Brouwer 1908, p. 109): in absence of any formula, no rule corresponding to a logical connective can be applied and thus the derivation cannot be carried on.

cannot really accept such explanation. First, the fact of using a multi-succedent calculus makes it difficult to decide which is the conclusion-formula of a derivation. Secondly, and more importantly, it is not in fact always clear which kind of absurdity is obtained by cutting a proof of $\vdash A$ with one of $\vdash \neg A$. In particular, if A is a tautology, then $\neg A$ is an antilogy and the empty sequent would be itself an antilogy. But, if both A and $\neg A$ are non-tautologies, then we are not in right to conclude that the empty sequent is an antilogy. We are thus in a situation where there exists different proofs of the empty sequent but we are not able to establish whether they are the same or not. Nevertheless, such a result would be possible to achieve if we disposed of a cut elimination procedure allowing to show that all these proofs of the empty sequent reduce to the same cut free proof. Now, Schütte's completeness proof implies as corollary the cut admissibility for the system LK_R . But this result is not an effective one. It just states that if we have a proof with cuts, then it exists a proof of the same sequent without cuts, but it does neither furnish an algorithm for transforming the proof with cuts into the cut free one, nor it tell us anything about the form of the cut free proof. It cannot then be used to establish identity of proofs results. If we want try to define a cut elimination algorithm for $\mathrm{LK}_R^{✠}$ we have to appeal to the admissibility of the structural rules of weakening and contraction. The problem is that the cut elimination procedure defined in this way is not local and it does not give any information about the evolution of the set of instances of the ✠ rule during the process of cut reduction. This latter point is particularly delicate because our analysis focuses exactly on the study of which set of instances of the ✠ rule are used in order to derive a particular sequent. If this set of instances changes during cut elimination, then our analysis is likely to fail.

5 Liberalizing syntax

We try to define here a framework that like $\mathrm{LK}_R^{✠}$ allows to define non logically correct proofs, but that at the same time possess an algorithmic cut-elimination procedure not perturbing the set of proper axioms, i.e. the set of instances of the ✠ rule. In general, our aim is to have a framework suitable both for allowing the generation of semantics from syntactical procedures and for assigning a really interesting computational interpretation to these procedures, not limited to the mere availability of a proof search algorithm. More precisely, we ask for a framework where the semantical role of axioms can be explained without any appeal to the set-theoretical notion of models, which is still based on a primitive and epistemic-transcending notion of truth. In order to do that we will try to define the notion of truth over that of proof but, differently from standard inferentialism – where what counts is only the order of applications of rules –, here proofs are analyzed with respect to their computational content. This perspective implies not to regard proofs as singular objects of study, but to consider them always in connection with some environment: if proofs correspond to programs, then their computational behavior can be detected only inside a context of evaluation.

What we need in order to define such a framework is to liberalize our syntax, in the sense that we don't want that the manner in which proofs are usually presented (i.e. as trees) affects our way to interpret them, i.e. as ordered sequences of inference rules. In fact, proofs can be viewed also from a geometrical point of view where the order of rules' application is not relevant, and what it

counts is only the "spatial" configurations of the premisses and conclusions of the rules and, what is more, the transformations that can be operated on these configurations keeping them invariant. The most appropriate objects for codifying such perspective are no more the syntactical objects inductively generated by a grammar, but are kinds of mathematical objects not necessarily representing ordered or inductive structures. Moreover, the operations definable over these objects have a computational content, so that the proofs-as-programs paradigm from which we have started is guaranteed.

Let us now present in some more details the perspective just sketched.

5.1 Proof nets and axioms

The best suited frame for developing the project of liberalization of syntax is, in our opinion, that of *linear logic* (Girard 1987b). First of all, it has to be noticed that adopting such a point of view does not mean to put into question the classical point of view that we have decided to defend in this paper. The reason is that linear logic is nothing else but a way to analyze classical logic at the microscope. Namely, by imposing a control on the use of structural rules of weakening and contraction –the latter corresponding to the decomposition of standard implication $A \to B$ into two distinct operations: a linear implication $\multimap$ and an exponential modality ! allowing the repeated use of the argument of type A – it is possible to let emerge from the set of rules for classical logic two sets of rules: the set of rules with shared derivational contexts – also known as *additives rules* – and the set of rules with independent derivational contexts – also known as *multiplicative rules* (see Di Cosmo & Miller 2001, §2.1).[14] In fact, for the purposes of our presentation we can limit our analysis to the multiplicative fragment of linear logic, **MLL**, which is composed by a closure operator $^\perp$ corresponding to an involutive negation (which will be specified later), multiplicative conjunction $\otimes$, and its De Morgan's dual, that is multiplicative disjunction $\parr$. Linear implication, instead, becomes definable exactly as in classical logic, i.e. $A \multimap B \equiv A^\perp \parr B$. The rules corresponding to these connectives are the following

$$\frac{}{\vdash P, P^\perp}\ \text{ax}$$

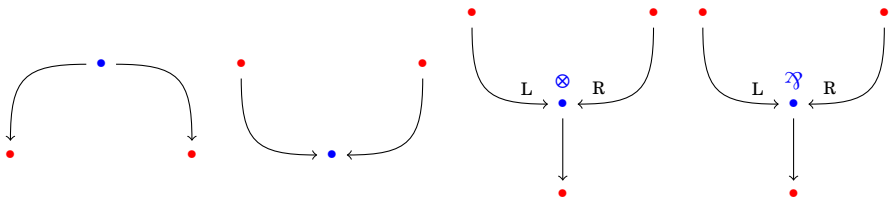
$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\ \otimes \qquad\qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\ \parr$$

and the cut rule is

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

What makes linear logic particularly interesting for our discussion is the way in which proofs can be represented, or better interpreted. Roughly speaking, the idea is to consider a kind of semantical entities which allow to deal with proof

[14] In fact, what we have just stated is correct only in the case of binary rules, while it is less clear in the case of unary rules. When there is only one premiss, there is only one context of derivation and hence the problem of sharing or splitting it seems not to make sense. However, when the immediate subformulas of the conclusion of a unary rule are all present in the premiss, we can consider it as an hint of the fact that in order to reconstruct the proof we need to follow all these formulas, since they are not derivable form the very same context. We conclude that the distinction we traced is not so ambiguous as we could have thought at first sight.



(a) Axiom Brick (b) Cut Brick (c) Tensor Brick (d) Parr Brick

Figure 4: Basic bricks for proof structures

systems analogous to multi-conclusion natural deduction. In order to do that, we have to abandon the syntactic and linguistic analysis of proofs and replace it with a purely geometrical analysis, where formulas are no more considered as linguistic acts, but as objets organized according to certain spatial relations over which invariant transformations can be executed. In other words, a formula is identified by the position it occupies and not by its syntactic form. It is on this idea that rests the notion of *proof nets* introduced by Girard (1987b). More precisely, a proof is represented by a graph constructed from asic elements representing the axioms, the connectives and the cut rule (see Figure 4). A graph obtained in this way is called a *proof structure*[15] and every sequent calculus proof can be represented as a proof structure, even though this correspondence is not injective. The non injectivity of this representation is the main motivation for the definition of proof structures which are meant to represent the quotient of sequent calculus proofs up to uninformative commutations of rules.

However, and that is where the main interest in proof nets lies, the syntax of proof structures is extremely tolerant and it allows to construct graphs that do not come from a sequent calculus proof. Hence, proofs (i.e. proof nets) are nothing but a particular subset of a set of more general structures (i.e. proof structures). Of course, this would not be helpful at all if we were not able to distinguish proof nets among proof structures: for this purpose, several correctness criterions exist, which distinguish proof nets as the set of proof structures satisfying given a geometrical or topological property.

5.1.1 Correctness Criterions

Many such criterions exists (see Seiller 2012c, §2.3 for a survey), but they all share the same global picture (Seiller 2012c, pp. 24-25). Given a proof structure $\mathscr{R}$:

1. we define a family S of objects – that we will call *tests*;
2. we show that $\mathscr{R}$ is sequentializable if and only if all elements in S satisfy a given property.

[15] The terminological choice adopted here is far from being accidental. On the contrary, it is intended to focus on the idea that a proof refers to a structure not only as a syntactical entity, but also as a semantical one, as we have seen in §3.3 and will clarify later, especially in §5.2.

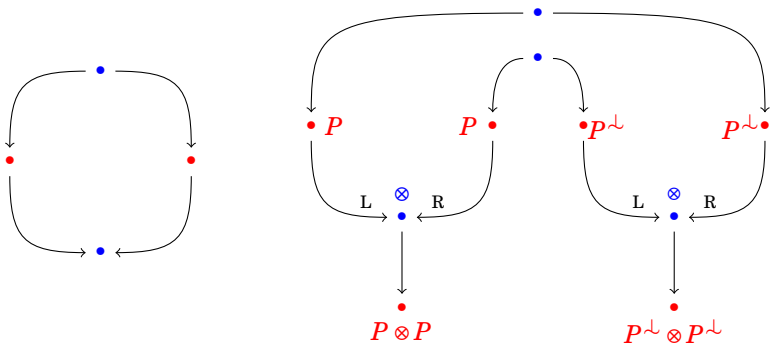
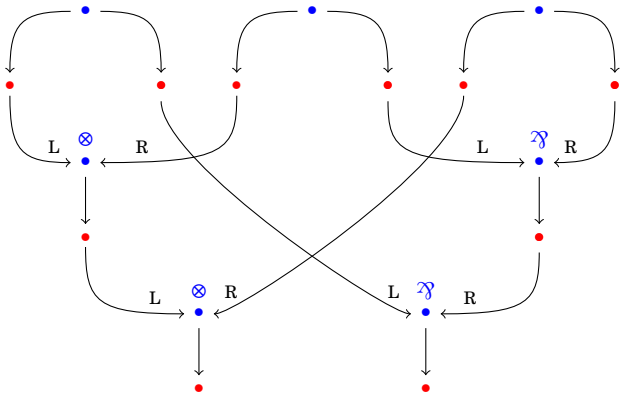


Figure 5: Proof Structures that are not Proof Nets

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash P, P^{\perp}}\,\text{ax} \quad \dfrac{}{\vdash Q, Q^{\perp}}\,\text{ax}}{\vdash P \otimes Q, P^{\perp}, q^{\perp}}\,\otimes \quad \dfrac{}{\vdash R, R^{\perp}}\,\text{ax}}{\vdash (P \otimes Q) \otimes R, P^{\perp}, Q^{\perp}, R^{\perp}}\,\otimes}{\vdash (P \otimes Q) \otimes R, P^{\perp}, Q^{\perp} \parr R^{\perp}}\,\parr}{\vdash (P \otimes Q) \otimes R, P^{\perp} \parr (Q^{\perp} \parr R^{\perp})}\,\parr$$

(a) Sequent Calculus Proof



(b) Proof Net

Figure 6: Proof of Associativity of the Tensor

The similarity runs deeper if we are a little more precise: in each case, the elements of S can be defined by the proof structure without its axioms, i.e. $\mathscr{R}$ in which we erased axiom vertices and their ingoing/outgoing edges. The second part of the criterion then describes how the axioms interact with this axiom-less part of the proof structure, which we will denote by $\mathscr{R}_t$.

Moreover, one can notice that the only difference between two proof nets corresponding to the same formula A will come from the axioms: the graph $\mathscr{R}_t$ being defined uniquely from A. Noticing this, one can then consider sets of axioms as *untyped proofs* and $\mathscr{R}_t$ as a type. The correctness criterion is then a simple typing criterion: if a set of axioms (an untyped proof) together with a type $\mathscr{R}_t$ yields a proof net, then this proof can be typed by the formula defining $\mathscr{R}_t$.

From now on, we will consider the correctness criterion based on the use of permutations (Girard 2011; Seiller 2102c):

- from $\mathscr{R}_a$ one defines a permutation σ_a;
- from $\mathscr{R}_t$ one defines the set of tests S as a set of permutations;
- $\mathscr{R}$ is a proof net if and only if for all $\tau \in S$, $\sigma_a\tau$ is a cyclic permutation.

We will say that two permutations σ, τ are *orthogonal* when $\sigma\tau$ is cyclic.

Notice the tests associated to $\mathscr{R}_t$ can be understood as counter-models. Indeed, if an untyped proof $\mathscr{R}_a$ cannot be typed by $\mathscr{R}_t$, it means that $\mathscr{R}_a$ is not a proof of the formula corresponding to $\mathscr{R}_t$. But the fact that $\mathscr{R}_a$ cannot be typed by $\mathscr{R}_t$ translates as the existence of a test in S such that the product of σ_a (the permutation associated to $\mathscr{R}_a$) and τ is not cyclic. Showing that an untyped proof $\mathscr{R}_a$ is not a proof of a formula A then boils down to finding a test of A that is not passed by $\mathscr{R}_a$, in the same way one should find a counter-model contradicting the set of axioms to show a derivation in $\mathrm{LK}^{\maltese}_R$ is not correct.

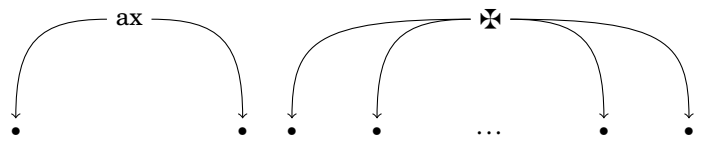
5.1.2 Cut elimination

Now, there exists a cut-elimination procedure on proof structures, and this procedure is actually compatible with the interpretation of proof nets $\mathscr{R}$ as a couple consisting in an untyped proof $\mathscr{R}_a$ together with a type $\mathscr{R}_t$. Indeed, this cut-elimination procedure is strongly normalizing and we can therefore choose particular strategies of reduction. Let us consider $(\mathscr{R}_a, \mathscr{R}_t)$ and $(\mathscr{P}_a, \mathscr{P}_t)$ two proof nets linked by a cut rule. We now consider the reduction strategies that first eliminate all cuts between $\mathscr{R}_t$ and $\mathscr{P}_t$, and then eliminate cuts between $\mathscr{R}_a$ and $\mathscr{P}_a$. This decomposition leads to the following interpretation:

- the cut elimination between types ensures that the specifications are compatible: if a cut cannot be eliminated then the strategy stops, which means that the two untyped proofs were not typed properly;
- the cut elimination between types, when successful, has no real computational meaning: it only defines a type $\mathscr{Q}_t$ and describes how the untyped proofs $\mathscr{R}_a$ and $\mathscr{P}_a$ are *plugged* together;
- the cut elimination between untyped proofs contains the computation, and yields an untyped proof $\mathscr{Q}_a$ such that $(\mathscr{Q}_a, \mathscr{Q}_t)$ is a proof net.

5.1.3 Generalized axioms

As it is the case in the framework described in Section 3, it is possible to extend the proof structures syntax by considering generalized axioms: In this setting,



(a) Identity Axiom (b) Generalized Axiom

Figure 7: Generalizing Axioms in Proof Structures

the generalized axioms represent a cyclic permutation, and allow one to prove any formula, as the daimon rule allowed to prove any formula in the system $\mathrm{pLK}^{\maltese}_{R}$. But the change of paradigm, from sequent calculus to proof nets, reinforced the rôle of these generalized axioms. Once again, we can write such a generalized proof net as a couple $(\mathscr{R}_a,\mathscr{R}_t)$ of a type $\mathscr{R}_t$ and a *paraproof* $\mathscr{R}_a$. Paraproofs are objects that do not necessarily contain correct instantiations of axioms and rules. In this setting, one can prove that there is a correspondance between paraproofs of a formula A and the tests of its dual $A^{\perp}$. Indeed, if $\mathfrak{P}(A)$ denotes the set of proof nets $(\mathscr{R}_a,\mathscr{R}_t)$ where $\mathscr{R}_t$ is the type corresponding to a formula A in this setting, then $\mathfrak{P}(A)$ corresponds to bi-orthogonal closure of the set of tests defined by the type $\mathscr{P}_t$ corresponding to the formula $A^{\perp}$.

The circle is now complete:

- an untyped paraproof can be given a type A if and only if it is orthogonal to the tests for A;
- an untyped paraproof is a test for the type A if and only if it is orthogonal to the proofs of A, i.e. proofs are tests for tests;

These remarks on proof structures allow to give more strength to the idea that generalized axioms are a way of adding counter-models to the syntax. As we will see in the next section, this idea can be even transformed in a redefinition of logic, where the objects are generalized untyped proofs, and formulas are defined interactively.

5.2 Untyped proof theory

These remarks lead to the definition of the first construction of a *geometry of interaction*, where basic objects are permutations. This construction was then generalized to take care of more expressive fragments of linear logic. We will here describe this type of constructions in a very general way – including in particular *Ludics* (Girard 2001), which is our main reference for the ideas developed in §7 –, and call such a framework an *untyped proof theory*.

Definition 29. An untyped proof theory is given by:

- A set of *untyped paraproofs*;

- A notion of execution;
- A notion of termination given by a set of untyped proofs Ω.

The idea is then to construct everything from the notion of execution. One first defines a notion of orthogonality: two paraproofs a, b are orthogonal – denoted $a \perp b$ – if and only if their execution $a :: b$ is in Ω. It is worth noting that no particular constraints are imposed on Ω. This means that the notion of termination is not an absolute one, but it depends from what we decide to consider as being a terminating configuration for the untyped proofs we are considering. Hence, a system of untyped proof theory represents an extremely flexible framework for developing a computational setting. On the other hand, from a more mathematical point of view, the notion of orthogonality corresponds intuitively to the phenomenon that occurs between generalized axioms in proof structures: if $a \perp b$, then a (resp. b) must be a paraproof of a formula A (resp. $A^{\perp}$), or equivalently a test for $A^{\perp}$ (resp. for A). Following on this parallel, one defines *types* as sets of paraproofs equal to their bi-orthogonal. This is actually equivalent to defining a type as a set of paraproofs A such that there exists another set of paraproofs B with $A = B^{\perp} = \{a \mid \forall b \in B, a \perp b\}$, i.e. a type is defined as a set of paraproofs that pass a given set of tests B.

With these definitions one can then define logical constants as a construction between paraproofs, a construction that then induces a construction on types. Several such constructions exist and allow the reconstruction of (sometimes only fragments of) linear logic.

Example 30. The first construction based on permutations mentioned at the beginning of this section can be enriched in order to give a construction for MLL with units (Seiller 2012b), for MALL with additive units (Seiller 2012a), and even for Elementary Linear Logic (Seiller 2013), a linear logic with constrained exponential connectives that characterizes the set of functions computable in elementary time. In this setting, the set of paraproofs is a set of couples of a graph and a real number, the notion of execution is based on the graph of alternating paths between two graphs, and the notion of termination is given by the set of couples $(a, \emptyset)$, where $a \neq 0$ is a real number and $\emptyset$ denotes the empty graph on an empty set of vertices.

Example 31. In Ludics, the set of untyped paraproofs is the set of *designs-desseins*, the execution is the cut-elimination procedure over these, and the notion of termination is the set of dessein containing a single design: the daimon.

Example 32. In the latest version of geometry of interaction, the set of untyped paraproofs is defined as the set of *projects*, while the execution is defined as the solution to the *feedback equation* and the termination is defined as the set of conducts of empty carrier with a non-null wager.

In these frameworks, one can then characterize which paraproofs correspond to proofs in the same way we defined correct derivations amongst the derivations of $\mathrm{LK}_R^{\maltese}$. These objects, styled *successful*, or *winning* to emphasize their relation to winning strategies in game semantics, can be tested against unsuccessful ones. The latter corresponding to counter-models, we are thus in a framework where the syntactical and semantical (in the classical sense) aspects of logic are both represented in a homogeneous way. In particular, the question of distinguishing between antilogies and non-tautologies is not anymore a question of verifying

each counter-model of a formula A, but of deciding if the set of tests of A, i.e. $A^{\perp}$, contains a successful paraproof or not.

6 Other similar frameworks

6.1 The idea of realizability semantics

As we have seen, in an untyped setting, proper axioms can be considered as sets of (not necessarily correct) paraproofs. We are thus far from the view of axioms as the starting point of derivations: they are rather interpreted as a particular kind of proof-structures. From this perspective, it becomes possible to look for the computational behavior of axioms.

The framework we presented has relevant analogies with what is know as Krivine's classical realizability. Both of these two frameworks are based on the notions of execution and termination applied over an untyped set of objects. Nonetheless, significant differences occur between these two approaches. In particular, Krivine's approach still consider the generation of untyped objects as tied to some ordered or inductive structures, as syntactical structures are. In order to point out these differences, we have to sketch the basic idea of realizability semantics, in general.

Historically, the notion of realizability was introduced by Kleene (1945), who was inspired by the finitist explanation of existential statements given by Hilbert & Bernays (1934, p. 32): a statement like $\exists x A(x)$ remains an incomplete communication until a finitary method is given for obtaining a witness t of the existential operator which allows to show $A(t)$ to be a complete communication (Kleene 1973, p. 5). The guiding idea is that this effective finitary method can be formally represented by a computable function; this induced Kleene to define a formal semantics based on recursive functions, that is a semantics based on essentially intensional objects, rather than on explicitly extensional ones, as it is the case of set-theoretical or relational semantics, like for example Kripke models. Indeed, recursive functions are nothing but algorithms, which are intentional objects *par excellence*. This becomes particularly clear when we consider Kleene's normal form theorem (see Sørensen & Urzyczyn 2006, p. 367), as it allows to represent every (total) recursive function by using its computation tree. This means that a recursive function is described not only with respect to what it does – i.e. by indicating the values it obtains when applied to certain arguments – but also with respect to how it acts – i.e. indicating the steps it accomplish in order to obtain the expected values. More precisely, Kleene's (number) realizability allows to associate every provable formula A of intuitionistic logic (resp. arithmetic) – proper axioms included – to a natural number n codifying (by a Gödel numbering) a recursive function f guaranteeing that A holds. In this case, it will be said that n realizes A – i.e. $n \Vdash A$ –, and since n is specific for A, we can consider it as a *truth-maker* for A.[16] Kleene's realizability could then be thought as nothing else but

[16]It should be noticed that the difference between realizability and usual set-theoretical or relational models rests on the fact that the latter do not offer a fine-grained analysis of what means for a formula to be true. Indeed, it is the whole set-theoretical or relational structure that renders true all theorems in an indistinguishable way. In other words, this kind of structures do not provide any truth-maker specific for each theorem considered, the consequence being that all theorems are identified at the semantical level, i.e. they all refer to the same semantic value. This fact seems then to lead to an impoverishment of the way of understanding mathematical theories.

a formal way to capture the notion of intuitionistic truth as corresponding to the possession of a construction – or better, a proof –, thus respecting BHK interpretation. This seems to be confirmed by the fact that Kleene's realizability allows to verify the disjunction and the existential properties for intuitionistic arithmetic. And since this is obtained by interpreting intuitionistic arithmetic over a fragment of arithmetic itself, then Kleene's realizability can be see as a formal and rigorous characterization of the BHK interpretation, namely by using the technique of *inner models*. However, considering Kleene's realizability as a definition of the predicate of intuitionistic truth, as Tarski's notion of satisfiability is a definition of the predicate of classical truth, would be a too hasty conclusion. The problem being that there exists natural numbers realizing formulas which are not provable in intuitionistic logic (reps. arithmetic). In particular, even if the realizability relation is sound with respect to the intuitionistic relation of derivability, it can be shown that is not complete. In particular, it can be shown that for every closed formula A, either A or $\neg A$ is realizable (Sørensen & Urzyczyn 2006, pp. 244-245). The reason is that if a formula A is not realized, this means that the type A is not inhabited by any recursive function f, and due to the fact that by definition $\bot$ has no realizers, then every function can realize $\neg A$.[17]

However, apart from the problem of representing or not a faithful account of intuitionistic truth, what must be retained for our discussion from Kleene's realizability is that in order to construct a model for intuitionistic logic (resp. arithmetic), it makes appeal to the notion of recursive function and this notion reveals to be broader than that of proof; as we have seen, the set of realizers is bigger than that of proofs.

In the same vein as Kleene's intuitionistic realizability, Krivine's classical realizability represents a way to build models for classical theories based on "computational objects". Let us see in details what this does mean.

First, it should be noticed that what is common to all realizability models – being them intuitionistic or classical – is the fact that they do not simply consider whether a formula is true, but they also taken into account the way in which such formula is made true, i.e. they make explicit the truth-makers of the formula. Secondly, a fundamental aspect that Krivine's classical realizability shares with Kleene's intuitionistic realizability is the fact that the number of realizers exceeds the number of proofs. But this does not imply, as in Kleene's account, the existence of formulas which are realizable without being provable; it means, instead, that the set of realizers of a formula does not contain only proofs, but also other kind of objects. This fact seems to confirm the idea that realizers can be seen as truth-makers, that is as the entities that *entail* – if not even *bring about* – the truth of a sentences. Proofs, on the contrary, seem in addition to *explain* why that sentence is true. Proofs do not simply entail the truth of sentences, but they represent the *grounds* for their truth.[18] Moreover, when we talk

[17] In fact, there are at least another aspect that provide from identifying Kleene's realizability with BHK interpretation. In the BHK interpretation the concepts «that figure in meaning explanations, [...] have to do with our cognitive capacities »(van Atten 2012, §4.5.2). In particular, the key concept of effecting mathematical construction should be conceived such that we recognize an effective mathematical construction when we see one. But this is not the case for Kleene's realizers: it is indeed undecidable whether as given number n realizes a certain formula A (Dummett 1977, p. 320; Sørensen & Urzyczyn 2006, p. 244-245).

[18] This distinction between truth-makers and grounds has been inspired by a remark made by P. Martin-Löf during the conference *Truth-Makers and Proof-Objects*, held in Paris at the École Normale Supérieure, on November, 23-25, 2011. It should be noticed that an analogous appeal to the

about truth-makers we are simply talking of posits, that is of entities that are not necessarily accessible to human agents, but that are nonetheless conceivable, in so far as they are logically possible.[19] Grounds, instead, have to be understood as evidences, in the sense that they are particular kind of entities that are not only accessible to human agents but they are also warrants for human knowledges. In other words, they are not only logically possible, but they are really possible.[20]

However, Krivine's realizability is not a simple extension of Kleene's realizability. Not only there exists intuitionistic valid formulas the truth-makers of which are not classical realizers (see Miquel 2009a, p. 79). But the most striking difference being the fact that the former leads to define realizers by exploiting an enlarged notion of program. This is because a program is no longer identified with a computable function on natural numbers. On the one hand, Kleene's realizability focuses only on input and output values, so that functions still keep a trace of a set-theoretical nature: they are eventually reducible to sets of pairs of natural numbers. In this sense, the domain and the codomain of computable functions are already fixed from the beginning (in both cases the set of natural numbers), and so computable functions have to be considered as typed objects. On the other hand, in classical realizability, programs are represented by objects that have deeper intensional features/nature: what matters is what the program does, and how it is executed in a given *context*. It is worth noting that this reference to a set of contexts represents a crucial step. The reason is that the behavior of programs is no more established in advance, that is to say, the programs forming the set of realizers are not *a priori* typed objects but they are only later assigned to a type on the basis of their computational behavior, which is in turn not defined in an absolute way, but depends from the given context. This agrees with what we already said in §5.2.

6.2 Classical realizability in a nutshell

This change of perspective confers a certain flexibility to Krivine's framework: classical realizability can be extended to classical logic and other mathematical theories exploiting the fact that to every proper axiom can be assigned a different term constant codifying a certain programming operation. For example, Peirce's law is taken to be the proper axiom distinguishing classical from intuitionistic logic and it is associated with a program instruction analogous to the `call-with-current-continuation` control operator of the programming lan-

term 'grounds' can already be found in Prawitz (2012a, 2012b). In general, these kinds of use of the term 'grounds' seem to refer to the term *Begründungen* – usually translated as 'justifications' – introduced by Bolzano (1817, p. 228) in order to denote the «presentations of the objective reason for the truth concerned». But since Bolzano opposes proofs as *Begründungen* to proofs as *Gewissmachungen* – that is proofs the purpose of which is simply to convince –, we also should be more precise in our terminology and specify that when we speak of proofs as grounds we are speaking only of normalizable proofs, since these are the proofs that can be put into canonical form. On the other hand, proofs of the same sentence A that cannot be normalized are no more than arguments used for convincing of the truth of A. In other words, they are only subjective proofs, and not objective ones, because they cannot be reduced to a common and unique manner of proving A. For more details about the connection between proofs as *Begründungen* and normal proofs see Cellucci (2004, §§9,10) and Dubucs (2003, p. 195).

[19] The expression 'logically possible' is used here as synonym of 'possible in principle', so that the term 'logical' is not used to refer to some specific formal system. For an analogous use see Sundholm (1994, p. 160 sqq.).

[20] For more details about the notion of a really possible entity see Sundholm (1994, p. 161).

guage SCHEME. Similarly, the axiom of countable choice is associated to a program instruction akin to the `quote` instruction of the programming language LISP (Krivine 2003). We will come back later to these examples.

The fundamental notion of computation on which Krivine's account rests is obtained using three key ingredients: *terms*, *stacks*, and *processes*. From a syntactical point of view, terms are composed by purely λ-terms enriched by two sorts of constants:

i) *instructions*, noted with κ, and ranging over a non-empty set $\mathscr{K}$, containing at least the constant `cc` which corresponds to the control operator `call-with-current-continuation`.[21]

ii) *continuations*, noted with k_π, and where π ranges over the set of stacks.

Stacks, instead, are *lists* of closed terms, the last element of which is the stack constant $\diamond$, standing for an empty stack. Terms and stacks are defined by mutual induction according to the following grammar:[22]

Terms	t, u	::=	$x \mid \lambda x.t \mid (t)u \mid \kappa \mid k_\pi$	$(\kappa \in \mathscr{K})$
Stacks	π	::=	$\diamond \mid t \cdot \pi$	(t closed)

As already mentioned, terms correspond to programs and their role is to be the truth-makers of sentences. From the morphological point of view, they can be divided into two categories: those that contain continuations and those that do not. A term containing no continuation constant is called a *proof-like* term. Intuitively, such a term corresponds to a (correct) proof, and thus its role is not just to be a simple truth-maker, but it can be considered as a ground.

Stacks, on the contrary, correspond to the evaluation contexts of programs, as they are the environments within which programs "react" and exhibit a specific behavior. Finally, processes are obtained by letting (closed) terms and stacks interact. Thus, contexts can be seen as *tests* for programs. Given a (closed) term t and a stack π, a process is noted by $t \star \pi$. Computation is then defined by exploiting an evaluation relation on processes, noted with $\rightsquigarrow$, and defined in the following way:

$$\begin{array}{rcllcrcl}
\lambda x.t & \star & u \cdot \pi & \rightsquigarrow & t[u/x] & \star & \pi \\
(t)u & \star & \pi & \rightsquigarrow & t & \star & u \cdot \pi \\
\mathtt{cc} & \star & t \cdot \pi & \rightsquigarrow & t & \star & k_\pi \cdot \pi \\
k_\pi & \star & t \cdot \pi' & \rightsquigarrow & t & \star & \pi
\end{array}$$

An examination of these clauses reveals that only the first one possibly corresponds to computing the value of a program, once that an input coming from the context is given. This is particularly clear looking at the limit case, when π is $\diamond$, since the process obtained is composed by a term and an empty stack $\diamond$

[21] The reason is that Krivine's realizability has been conceived for investigating classical logic, and a way to obtain classical logic from intuitionistic logic is to add the Peirce law $((A \to B) \to A) \to A$, which as we mentioned corresponds to the operator of `call-with-current-continuation`.

[22] It is worth noting that Krivine's classical realizability has been mainly conceived for second order theories. For this reason, among the set of terms the operators of pair construction, projection, injection, and case analysis do not appear: at the second order level they become definable (see Sørensen & Urzyczyn 2006, pp. 280-281).

and thus there is no other term to be taken from the context and used as an input. On the contrary, in all the other cases, even when π is $\diamond$, the execution does not bring to a process with empty context, but it construct a new context. The other clauses thus establishes how programs and context can mutually modify one another. The consequence is that, differently from the standard intuitionistic proofs-as-programs approach, in Krivine's approach the evaluation process does not necessarily compute a value, but it simply has to guarantee that a program terminates when used in some specific contexts. This fact suggests to interpret the evaluation contexts as *false-makers*, that is as counterexamples that when opposed to the corresponding truth-makers they produce a deadlock, i.e. something corresponding to a sort of antinomic situation.[23] Moreover, it has to be noticed that in analogy with the untyped setting, the notion of termination is not an absolute and unchangeable one, but it depends on which processes configurations have been chosen to represent the terminating states, or better, the terminable ones. In other words, a program t is considered to be terminating with respect to a context π when $t \star \pi \in \perp\!\!\!\perp$, for any arbitrary set of processes $\perp\!\!\!\perp$ closed under anti-evaluation. This last condition meaning that, given two processes p and p', such that $p \rightsquigarrow p'$ and $p' \in \perp\!\!\!\perp$, then $p \in \perp\!\!\!\perp$ obtains.

By comparing these notions of execution and termination with those presented on page 22, it becomes quite natural to look at contexts as generalized axioms, and terms containing continuations as paraproofs. We will return to this analogy later, but for the moment we can already notice that since the notion of computation is not reduced to the the fact of obtaining or not a value – by appealing to some mechanical means –, then also the notion of realizability is not reduced to the simple fact that a type is inhabited or empty. Thus, in order to better appreciate this notion of computation we have to understand the semantical role it plays, especially in defining a notion of meaning compatible with classical logic.

6.3 A computational account of meaning

If we turn to a conceptual analysis of Krivine's framework, it can be immediately noticed the distance dividing it from the Dummettian inferentialist theory of meaning sketched in §1. The main reason is that the understanding of a sentence A is no more based only on a single semantic notion, that of proof of A – or more precisely, that of canonical proof of A – but it requires to make appeal to two semantic notions at the same time: programs – corresponding in a loose way to proofs – and contexts. These two notions are one the complement of the other, so that a special kind of bivalence is introduced at the semantic level: every well-formed object belonging to the realizability level corresponds either to a program or to a context. Thus, differently from the verificationist perspective, if a sentence does not semantically correspond to a set of proofs sharing certain common

[23] Strictly speaking, these antinomic situations do not imply the incoherence of the system itself. The reason is that, as we already mentioned, truth-makers, as well as false-makers, are only posits. In this sense, it is not astonishing to conceive two logically incompatible situations together: the resulting conflict between these two situations would be only a conflict in principle, not an actual one. On the contrary, a genuine incoherence is obtained when two contrary evidences present, namely when it is possible to exhibit two proofs of two opposite propositions, respectively (see Miquel 2009a, p. 81). This way of understanding incoherence is the same professed by Hilbert: incoherence was definable only at the level of «concrete objects» (Hilbert 1926, p. 376), i.e. at the level of finitary arithmetic, not at the level of logic.

properties, it does not mean that this sentence has no semantical counterpart at all and that its corresponding type is empty. On the contrary, its semantic correlate will be a set of contexts sharing certain common properties. In this sense, even if semantic values for sentences are not taken as primitive, like in standard Tarskian semantics – since they have been defined over the primitive notions of program and context –, and even if they are more in number than the usual two semantic values adopted by standard Tarskian semantics – since for every formula there exists a semantic value specifically corresponding to that formula –, nonetheless a kind of bivalence is induced by the binary partition operated by the computational approach distinguishing between programs and contexts. This very fact represents a change of perspective with respect to the standard verificationist account: the main focus is no longer on proofs *per se*, but on the interaction between programs and contexts. In particular, the introduction of contexts opens a way to assign an operational meaning to proper axioms, without needing to appeal to the inferential aspects characterizing proofs. In fact, from the verificationist point of view, assigning a semantic value to proper axioms is a discouraging task because assigning a set of canonical proofs to proper axioms is a sort of a counter sense, since by definition proper axioms are sentences accepted as true without any specific proofs to be exhibited. On the contrary, in Krivine's account, the meaning of a proper axiom is not given on the basis of its inferential behavior, but on the basis of its computational behavior. The latter not being defined in absolute terms, but with reference to a given set of contexts, that is to those situations which oppose to the axiom and try to falsify it.[24] The idea is thus to identify those entities which make the axiom true in spite of all possible attempts made to refute it. And even if by definition this entity cannot be a proof, it remains nonetheless an entity accessible to human agent, since it corresponds to an operation consisting in an algorithmic manipulation of a given set of syntactical objects, where the latter represent the context of evaluation. Under this perspective, it seems not to be an exaggeration to say that Krivine's proposal respects in some specific sense the spirit of Hilbert's finitist program. More precisely, like in Hilbert's account, sentences are meaningful only when they can be associated – or somehow reduced – to concrete objects or to finitist operations defined over these objects. In particular, in the realizability setting, standard Hilbert's strokes |, ||, |||, etc. are replaced by terms and contexts, and since terms and contexts are syntactical objects, that is nothing else but finite configurations of signs, they are also objects existing in time and space (see Martin-Löf 1970, p. 9); they can thus be considered as concrete objects as well as Hilbert's strokes.[25] On the other hand, in order to show that Krivine's realizability can also recover the notion of finitist operation, the usual identification of this notion with that of primitive recursive function, as proposed by Tait (1981), has to be abandoned and replaced with Kreisel's idea according to which the class of finite operations corresponds to the class of provably recursive functions in arithmetic (Kreisel 1960). Notice that this was exactly the class of functions that Kreisel explicitly characterized in establishing his *no-counterexample interpretation* (Kreisel 1951, 1952). And the no-counterexample interpretation, as stated by Krivine (2003, p. 260),

[24] Notice the analogy between contexts and countermodels, and compare it with what we said in §5.1.3.

[25] Actually, as Parsons remarked, Hilbert's strokes, as well as syntactical objects, are quasi-concrete objects: they are a particular kind of types, the «intrinsic [property of which is] to have instantiations in the concrete » (Parsons 2008, p. 242).

is actually the method inspiring his own analysis of the computational content of arithmetical theorems (see Bonnay 2002 for more details): when a sentence is provable, it is shown that is possible to extract an effective procedure capable of falsifying every possible counterexample for that sentence. However, this does not mean that Krivine's realizability could be eventually reduced to be a simple variant of Kreisel's no-counterexample interpretation. As we already mentioned, Krivine's realizers are not "ready-typed" functions but untyped programs. And since programs are intensional objects *par excellence*, they are assumed to convey much more information rather than functions.

First of all, differently from functions, programs are not reducible to some particular kind of step-by-step set-constructions on of natural numbers, but they aim at expressing every kind of actions effectively performable on syntactical objects in general, even on those that would not be considered in principle as well-typed. It is this general character that allows to assign a computational content even to those formulas that do not corresponds to theorems or axioms, and by showing that it is possible to assign them truth-makers. More precisely, these truth-makers are terms containing continuation constants. This means that the truth of a formula is guaranteed under the assumption that some counterexample for another formula is given. For example, a formula $A \to B$ can be made true not only by a proof of it, but also if a counterexample to A, *alias* a stack, is given. But since the information present in a stack π can be codified by a continuation k_π, and since a continuation is a term, then it is plausible to think this term as typeable with $\neg A$. Thus, modulo a certain degree of approximation, we can think that $A \to B$ is obtained using the following derivation in a natural deduction setting:[26]

$$\dfrac{\dfrac{\dfrac{\dfrac{k_\pi : \neg A \qquad [x : A]^{1.}}{(k_\pi)x : \bot}\ \to \text{elim}}{(k_\pi)x : \forall X.X}\ df}{(k_\pi)x : B}\ \forall^2 \text{ elim}}{\lambda x(k_\pi)x : A \to B}\ \to \text{intro (1.)}$$

The approximations that we are making here mainly consist in working as if terms are already typed, while in Krivine's framework untyped objects are considered. In order not to lose the greater expressivity allowed by working in an untyped framework, we have to license operations on terms containing free variables (in this case the application of k_π to the variable x). Moreover, our type assignment to k_π is not obtained by exploiting the instruction cc, but it is directly extracted form the intuitive reading of the role played by a context π, i.e. that of a falsifier. However, we can recover Krivine's reading of k_π once a proof of A is effectively given, i.e. when x is substituted by a closed term u. Looking then from the structural point of view, it should be noticed that $k_\pi : \neg A$ cannot be considered as a dischargeable hypothesis, since k_π is by definition a close term and not a variable. But it cannot be considered a closed premiss either, since it

[26]The deduction system adopted here is described in Miquel (2009a, p. 85). The idea is that by working in second order logic we obtain a *polymorphic* type system, that is a system where terms could be associated to more than one type. Since in this paper we adopted the convention to present terms in Curry style, this means that the information concerning types is not present in the terms, and thus polymorphism is not explicitly manifested inside terms – by means of some abstraction operator –, but remains implicit (see Hindley & Seldin 2008, p. 119-120). It is for this reason that the rule $\forall^2$ elim is not associated to any new operation on terms.

has not been justified by a proof, but it comes from the "reification" of a context. In a certain sense, the role played by $k_\pi : \neg A$ is that of a pretension, namely the pretension to accept that it is the case that $\neg A$, that is to work *as if* $\neg A$ has been proved. This is nothing else but considering that $k_\pi : \neg A$ plays the role of a *postulate* in the Aristotelian sense, i.e. as «something [which is] not in accordance with the opinion» of the person to whom the discourse is addressed (Aristotle, *Posterior Analytics*, 76b32-34; Barnes 1993, p. 16, 141-142). This seems to corroborate the idea that a continuation constants plays the same role of a test (see §5.1.1), and since tests are paraproofs, the conceptual reading of paraproofs is that of postulates (in the Aristotelian sense).

A second aspect we want to focus on is that programs do not simply tell us what can be computed, but also the manner in which the computation is performed. In particular, while two functions computing the same values (in correspondence of the same arguments) are identified, since they have the same graph, two programs giving the same outputs (in correspondence of the same inputs) are not, because it can be the case that they corresponds to two different algorithms. Thus, it can happen that two programs compute the same function without that we can identify them. Using programs it is then possible to differentiate two sentences on the basis of their computational content, which would have been otherwise identified, if we had used functions.[27]

6.4 Computational features

Classical realizability, is thus based on the central notion of computation, and the notion of computation in its turn on that of execution. The notion of execution is not a stable one, though. In particular, by adding a new proper axiom to the system, a new model is obtained, because a new constant instruction is added. And when a new instruction is added also the notion of execution has to be redefined. Consider, for example, the axiom scheme of countable choice, according to which every countable family of non-empty sets has a choice function. Written in the language of set-theory it takes the form:

$$\forall x \in \mathbb{N} \exists y \in S . A(x,y) \to \exists f \in S^{\mathbb{N}} \forall x \in \mathbb{N} . A(x, f(x)))$$

But when we want to add it to second order arithmetic $\mathbf{PA}^2$, we can simply write:

$$(\mathbf{ACC}) \quad \forall x \exists Y A(x,Y) \to \exists Z \forall x A(x, Z(x))$$

where Y is a k-ary second order variable, Z a $k+1$-ary second order variable, and $A(x,Y)$ is any arbitrary formula not containing Z free. The system $\mathbf{PA}^2 + \mathbf{ACC}$ is a theory adequate enough to formalize analysis. In order to realize **ACC**, and thus construct a realizability model for this theory, a new instruction χ has to be introduced, behaving in the following way (see Krivine 2003, p. 271):

$$\chi \quad \star \quad t \cdot \pi \quad \rightsquigarrow \quad t \quad \star \quad \underline{n_t} \cdot \pi$$

[27] A concrete example of this indifference to algorithms which is implicit in a function-based approach can be seen exactly in the definition of the class of provably recursive functions in arithmetic, where to establish that a function is provably recursive it is sufficient to show that *only one* of its algorithms is provably recursive (see Sørensen & Urzyczyn 2006, p. 239).

where $\underline{n_t}$ is the Church numeral[28] corresponding to the natural number n_t which is the number that has been associated to the term t by a (not necessarily recursive) enumeration of the set of closed terms. When this enumeration it is a recursive one, then χ can be implemented by means of the `quote` instruction of LISP.[29] Still, the crucial point to be noticed is that the introduction of χ induces a modification on program execution, since a new evaluation clause has to be considered and new contexts of evaluation can be created by exploiting the function enumerating closed terms. But in order to have a full characterization of the execution, it has to be checked whether the new evaluation clause remains compatible with previously defined program transformations, in particular with β-reduction. Actually, this is not the case for the χ operator (see Krivine 2003, p. 271). This means that χ leads up to differentiate terms that otherwise would have been identified. Hence, the identity criterion for computational entities not only rests on their behavior rather than on some pre-fixed features like their nature or form, but it also strictly depends on the situations in which this behavior is manifested. More precisely, when new instructions are introduced, and the context of computation changes, the behavior of terms could change as well. From a philosophical point of view, this phenomenon seems to support an anti-essentialist point of view, according to which an entity is recognized to belong to a certain category of objects not because it possess a fixed set of characteristic properties, but because we can use it for performing certain kind of operations. From the technical point of view, on the contrary, since β-reduction is strictly connected to the notion of proof normalization, its incompatibility with certain kind of instructions suggests that normalization has no more a central role within the realizability framework. But since proof normalization is at the core of the possibility of obtaining canonical proofs, this situation seems to confirm the idea that Krivine's approach is conceptually distinct from Dummett's one.[30]

[28] A Church numeral is a representation in pure λ-calculus of natural numbers, such that a given natural number n corresponds to the λ-term

$$\lambda f \lambda x \underbrace{(f)\ldots(f)}_{n \text{ times}} x$$

For more details see Sørensen & Urzyczyn (2006, p. 20).

[29] Notice that χ does not directly realize **ACC**. What can be proved instead is that there exists a function $F : \mathbb{N}^{k+2} \to \wp(\Pi)$, with $\wp(\Pi)$ the power set of the set of stacks Π, such that:

$$\chi \Vdash \forall x(\forall y(Nat(y) \to A(x, F(x,y))) \to \forall Y(A(x,Y)))$$

where $Nat(y) \equiv \forall X(X(0) \wedge \forall x(X(x) \to X(sx)) \to X(y))$. It is then easy to show that the term $\lambda z(z)\chi$ realizes what can be called the *intuitionistic countable choice axiom*:

$$\textbf{(IACC)} \qquad \exists U \forall x(\forall y(Nat(y) \to A(x, U(x,y))) \to \forall Y A(x,Y))$$

where Y is a k-ary second order variable, U a $k+2$-ary second order variable, and $A(x,Y)$ is any arbitrary formula not containing U free. In order to realize **ACC** it is sufficient to show that **ACC** can be obtained from **IACC** by means of logical equivalence, the least number principle, and the principle of extensionality of functions (see Miquel 2009b, §§8.1, 8.2.). It is in performing these deductive steps that an essential appeal to classical logic is made.

[30] In fact, the abandon of the notion of canonical proof as a semantic key concept was already accomplished at the moment of considering that the realizer of the Peirce's law is the constant instruction `cc`. The reason is that a constant corresponds to a one step proof, namely a proof obtained by applying a 0-ary rule, so that any possible analysis of the structure of this proof becomes pointless. A similar kind of blindness to proof structure is advocated by Kreisel (1951, pp. 155-156, note 1) in comparing his unwinding program to the Brouwer's constructivism.

However, the sensibility of execution to the addition of new program instructions prevents classical realizability from being a *uniform* framework for the treatment of axioms. This represents a major difference with respect to the untyped framework presented above, where the presence of two peculiar ingredients contribute to guarantee the possibility of working with a general and unique notion of execution. On the one hand the appeal to the ✠ rule allows to define a general notion of axiom, subsuming all kinds of axioms, being them proper axioms or identity axioms. On the other hand, no real distinction is made between terms and sacks. From the point of view of the untyped proof theory, both of these two kinds of entities correspond to paraproofs. Their differences being not fixed by syntactical features, as in Krivine's account, but only by the point of view which is adopted on them, or more precisely, by what is done with them. In this sense, the untyped framework seems to go even a step further than Krivine's realizability in the criticism of essentialism. We will come back to these themes in the next section, especially in §7.1.

7 Philosophical considerations

Hitherto we presented some ideas for developing a computational account which is general enough to allow the justification of logical statements as well as of proper axioms. However, our aim is not simply to present such an account as a mere technical device, but to show that the computational setting can constitute an appropriate framework for the development of an uniform and epistemic-based understanding of logic and mathematics. In order to do this, a linguistic point of view will be adopted, principally based on the analysis of the meaning of logical and mathematical sentences. Notice that our analysis will mainly focus on the untyped framework presented in §5, while for a somehow similar analysis of Krivine's realizability we refer the reader to Bonnay (2007).

7.1 Normative *vs*. descriptive theories of meaning

As we said in §1, in this paper we made an attempt of reconciling a standard notion of axiom with a certain kind of inferentialism based on the computational interpretation of proofs, or better, of proof structures. However, we have not yet clarified the philosophical extent of this kind of inferentialism. In particular, we have not yet made explicit which kind of theory of meaning can be induced by this computational perspective. Our claim is that such a theory of meaning is rather different from the one associated to standard inferentialism; the main difference being that the latter is *normative* while the former is *descriptive*. Let us try to clarify this crucial point.

Standard inferentialism is usually identified with Dummett-Prawitz verificationism (cf. Tennant 2012, §2). According to this approach, the rules governing our linguistic practice – i.e. the rule we are supposed to master in order to successfully perform linguistic exchanges – have to be submitted to a principle of harmony imposing not to generate new informations in a non-conservative way. Usually, this principle is formally captured by the so-called Prawitz's *inversion principle*: whatever follows from the assertion of a certain (complex) sentence A cannot exceed what follows from the direct grounds for asserting it, i.e. from the premisses of the introduction rule for A (Prawitz 1965, p. 33; Prawitz 1973).

This principle plays the role of a *norm*, in the sense that it transcends the situation that it regulates: by imposing it from the beginning of the construction of a language, it should be possible to guarantee *a priori* a somehow "perfect" communication, avoiding misunderstandings as well as other linguistically pernicious situations (cf. Dummett 1973a, p. 454, for the well known example of 'Boche').

The computational perspective we adopted in this paper shares with standard inferentialism the fact of assuming proofs as the meaning conferring objects. However, there is an essential difference between the two perspectives. As we already remarked, the computational point of view asks for the presence both of programs and of contexts into which the behavior of programs can be evaluated. Under the proof-as-programs correspondence this means that even if proofs are necessary in order to determine the meaning of a sentence, they are not yet sufficient. In particular, it is not sufficient to know the order of rules' application inside a proof in order to establish the latter as a meaning conferring object; we need also other inferential objects to play the role of contexts of evaluation. In the perspective we presented this is achieved by using the notion of para-proof. As we have seen, paraproofs are objects that do not necessarily represent correct – i.e. logically valid – (linguistic) arguments: the correctness of a para-proof depends on the interactional properties it displays in presence of other paraproofs. From the linguistic point of view, if a proof corresponds to a correct justification for the *assertion* of a a sentence, i.e. for judging that sentence as true (cf. Martin-Löf 1987; Sundholm 1997), a paraproof can be seen as an *argument* supporting the *utterance* (see Lecomte & Quatrini 2011a) of a certain sentence in a particular context of discourse, regardless of the fact that the argument is (logically) correct and the sentence is true. In the same vein, the process of interaction between two paraproofs can be seen as a dispute between two speakers both of them using arguments in order to convince the other to accept their own opinions. In this sense, truth is no more an absolute notion, but an interactional and "social" one: a sentence can be judged as true when we *always* dispose of an argument to convince the other speakers to accept it, i.e. when we dispose of a winning strategy. A further fundamental feature of this setting is that the meaning of sentences is not fixed by a set of rules that have to respect a pre-established principle, but it is determined inside the linguistic activity itself: to know the meaning of a sentence corresponds to know which arguments can be opposed to it in order to close the dispute, and this cannot be established in advance and "outside" the linguistic exchange itself because it strictly depends from the specific context and situation considered, in particular it depends from the arguments used by the other speaker. In this sense the untyped approach induces a sort of game-theoretical semantics. However, differently from standard semantics of this kind (see Hintikka 1983; Lorenzen & Lorenz 1978), to know the meaning of a sentence does not correspond to know how to gain the dispute. It is only requested to terminate the dispute; whether it is with a gain or with a loss this does not matter. As we mentioned before, the possession of a winning strategy corresponds to know that the sentence is true. Thus, sentences' meaning neither coincides with a truth-definition nor depends on a primitive and non-analyzed notion of truth. It is for this very reason that, in Dummettian terms, the computational and untyped approach can be characterized as an anti-realist position: «[...] the notion of *truth*, considered as a feature, which each mathematical statement either determinately possesses or determinately lacks, [...] cannot be the central notion for

a theory of the meanings of mathematical statements», on the contrary, «[. . .] it is in the mastery of [a] practice that our grasp of the meaning of the statements must consists» (Dummett 1973b, p. 225). In particular, characterizing the truth of a sentence as the possession of a winning strategy allows also to respect Dummett *manifestability* requirement (Dummett 1973b, pp. 93-95; Dummett 1976, pp. 79-82 ; Dummett 1977, pp. 193-195): the fact that a sentence is true makes a perceptible difference at the level of our linguistic practice, since there is someone that when argues in favor of it can always gain the dispute.

To sum up, the paradigm of this kind of computation-based theory of meaning is still the one according to which "meaning is use", even if in this case the set of the licensed uses of a sentence is not delimited by an absolute and external norm, but by the dispositions of the other speakers to reply to those uses. In this sense the notion of correct use of a sentence is detected inside the linguistic practice itself. Differently from standard inferentialism, it is held that the enterprise of a theory of meaning is not to fix *in abstracto* the rules that a language has to respect in order to work properly, i.e. in order to do what we expect it to do, namely to allow communication between speakers. In analogy with the position endorsed by Wittgenstein in the *Philosophical Investigations*, we can say that the enterprise is not to determine the «essence of a language» (Wittgenstein 1953, §97), that is not to determine the whole set of characteristic features that an abstract and idealized (concept of) language should posses. On the contrary, by the mere fact of existing[31], and of allowing communication between people, the linguistic activity has to be considered as something that already works properly and not as something that should be rectified (Wittgenstein 1953, §98). From such perspective, linguistic ambiguities - which are usually considered as sources of possible misunderstandings - are themselves considered as proper parts of the linguistic activity, instead of being rejected as incorrect.[32] This is a natural consequence of the absence of any *a priori* principles conceived for distinguishing between what is correct and what is not. In other words, the idea is that the rules governing our linguistic activity are *immanent* to it. On the one hand, this means that we become aware of the way in which meaning is assigned to sentences by describing the linguistic activity itself; on the other hand, the knowledge of the meaning of a sentence is manifested in the capacity of taking part to a linguistic exchange when this sentence enters into play, without this implying to make completely explicit the rules governing the linguistic exchange – otherwise an externalist approach would be adopted and not an immanent one, as we assumed to do.

7.2 Feasibility and interaction

The computational perspective we analyzed here can be considered as compatible with an inferentialist point of view as long as the application of an inference rule within a paraproof corresponds to (successfully) perform a linguistic act inside a linguistic exchange. However, what is peculiar to this computational perspective is that the choice of the applied rule is limited by the type of linguistic acts previously performed both by the speaker and by her opponent. In other words, the choice of the rules applied by the speaker strictly depends from the specific

[31]This fact can be established on the basis of an "empirical" experience of it.

[32]Indeed, in Ludics it is possible to represent fallacies in a formal way as shown by Lecomte & Quatrini (2011b).

linguistic situation she is confronted with. This has two main consequences. On the one hand, as we mentioned before, inference rules has to be considered as acting on linguistic objects that are situationally dependent, as utterances, and not on some more abstract or "absolute" linguistic entities like assertions (see Lecomte & Quatrini 2011a).[33] On the other hand, the fact that linguistic acts are situation-dependent means that the inference rules used in order to perform them have to take into consideration the particular type of resources available from situation to situation. Those considerations become evident when we think that the logical rules justified by the untyped setting are those of linear logic, since it is rather well acknowledged that linear logic is the clearest example of a resources-oriented logic (Di Cosmo & Miller 2010). Even if the computational untyped setting is characterized by anti-realistic features, as the fact of founding a theory of meaning on the possession and manifestation of some linguistic competences, rather than on a primitive and non-analyzed notion of truth, it still maintains some differences with respect to the standard form of anti-realism endorsed by Dummettian verificationism. In particular, it does not hold that mastering the meaning of a sentence consists in knowing what can be done *in principle* with that sentence, but it consists instead in knowing what can be *practically* done with it in some particular situations. In this sense, accepting a computational untyped approach yields to accept a sort of *radical anti-realist* position: to know the meaning of a sentence implies to know what can be *feasibly* done with it during a concrete linguistic exchange. We noticed, indeed, that the computational approach does not take into account idealized situations built by following an *a priori* fixed principle, but it focuses on already existing dialogical situations: its target is not to determine the principles necessary for the construction of a language, but instead to *represent* a language. Moreover, it is worth noting that these feasibility features are not acquired by imposing on verificationist's principles a further *a priori* constraint concerning proof size bounds,[34] for example by imposing a polynomial growth of proof length during normalization or cut-elimination obtained by changing usual connectives with linear ones (see Dubucs 2002; Dubucs & Marion 2003).[35] Actually, this change of connectives would be

[33]Moreover, as we already remarked, paraproofs are not necessarily correct and thus they are not connected to the notion of truth. An assertion, on the contrary, according to the Bolzanian tradition takes the form of the judgment 'A is true' (where A is a sentence or a proposition), and thus it is defined with respect to the very notion of truth, whether this is a primitive notion, as it is the case for realist positions, or whether it is not, as it is the case for anti-realist ones.

[34]These kinds of bounds are essentially dictated by two reasons: 1) guaranteeing that the process of *verification* that something is a proof can be practically done by human beings; 2) guaranteeing the semantic key objects, i.e. canonical proofs, to be objects that can be practically *constructed* by human beings. By respecting these two conditions it should be assured that, in the verificationist account, both truth and meaning never make appeal to entities transcending concrete human capacities, as it could be the existence of proofs the size of which goes beyond physical limits.

[35]The standard justification for the choice of polynomial bounds can be found in Wang (1981, §6.5) and it has been well summarized by Marion (2009), p. 424:

> It is generally agreed that polynomial-time computability captures the capacities of digital computing machines, as opposed to their *idealized* counterparts, the Turing machines. Digital computing machines do *not* have access to unlimited resources, and this seems to be the key point for a radical anti-realist program. It is only asked here from the radical anti-realist that she grants that digital computing machines are an unproblematic extension of human cognitive capacities, so that, with polynomial-time computability, one remains within the sphere of what is humanly feasible.

In fact, it seems to us that there is a further, and usually neglected, argument supporting this choice. Schematically, it can be presented in the following way: i) for the verificationist approach meaning is

justified only by an *ad hoc* reason. On the contrary, the bounds guaranteeing that the knowledge of sentences' meaning is based on linguistic skills that are feasibly manifestable is assured by the very nature of the interactional approach: the presence of two speakers, instead of only one, guarantees that the actions of one speakers are always bound by the actions of the other one, and *vice versa*.

7.3 Holism and molecularism

Our last observations will concern a typically Dummettian theme, namely the debate opposing a molecularist account of meaning to a holistic one.

Generally speaking, it is usually reckoned that the adoption of an axiomatic approach comes with the acceptance of some kind of holistic position (see Troelstra & van Dalen 1988, pp. 851-852). More precisely, presenting a theory in an axiomatic way has two main consequences with respect to our understanding of the set of sentences constituting the theory itself. On the one hand, the syntactical behavior of the expressions composing the language of the theory is fixed only by considering them all together: an expression is defined on the basis of the relations it entertains with the other expressions, and there is no *a priori* bound on the number of expressions that can be mutually related by the axioms. On the other hand, the inferential behavior of an axiom can be fully determined only when it is used in conjunction with other formulas in order to extract some relevant information from it, and also in this case no *a priori* bound can be imposed on the set of these formulas.[36] This situation seems to openly conflict with the molecularist approach defended by Dummett and according to which, in order to understand the meaning of a sentence, it must be required to master only a limited and well determined fragment of a language and not its whole totality (Dummett 1976, p. 79). But as our aim is to try to defend an axiom-based point of view, it may be wondered whether our approach has to be considered essentially holistic or whether at last it may be rendered somehow compatible with a kind of molecularism akin to the Dummettian perspective.

As a first general observation, it should be noticed that if on the one hand the untyped computational perspective allows to defined types – and thus also formulas and sentences – as sets of paraproofs (cf. §5.2 *supra*), on the other hand it does not provide a standard inductive definition of them; in particular, there is no such notion as that of *atomic type*. At first sight, this seems to contrast with Dummett's advocated molecularism, as far as the letter presupposes the possibility of ranking sentences in a hierarchy according to their increasing complexity.[37] In absence of a way to fix such a hierarchy of types-formulas, it could be the case that we are not able to assign a bound on the complexity of those formulas to

based on proofs, and the only logical rules allowed for constructing these proofs are the intuitionsitic ones; ii) via the Curry-Howard correspondence each proof of intuitionistic logic can be associated to a computable function, and *vice versa*; iii) a fundamental property of a theory of meaning is compositionality; iv) by restricting to polynomial computable functions, compositionality between functions is preserved: the composition of two polynomial computable functions (i.e. intuitionistic proofs) is still a polynomial computable function (i.e. intuitionistic proof).

[36]The other way round, this situation corresponds to the idea that to understand the meaning of an axiom it is necessary to understand the *totality* of the consequences that can be drawn from it (see Dummett 1991, p. 228).

[37]In particular, see Dummett (1991, p. 223): «Compositionality demands that the relation of dependence imposes upon the sentences of the language a hierarchical structure deviating only slightly from being a partial order ».

which we have to make appeal in order to understand another given formula. More generally, as we already said, in the framework of a computational perspective, to know the meaning of a sentence corresponds to be able to take part to a linguistic exchange once this sentence makes its appearance. It seems then easy to think that if there is no way to fix in advance the kind of formulas that will be involved in the exchange, then, in order to explain the understanding of the given sentence, a possible appeal to the understanding of the whole language could be required. This seems to mark a relevant difference with Dummettian verificationism. Let us try to sketch a tentative explanation.

According to the Dummettian point of view, understanding the meaning of a complex sentence A can be reduced to understand the meaning of its principal connective and, in order to do it, to take into consideration only the set of sentences in which this connective appears as the principal connective. In this manner only a limited fragment of the language has to be analyzed in order to know the meaning of a certain expression. More precisely, this kind of analysis can be essentially done by focusing on the properties of the inference rules involved in the justification of the direct assertion of the sentence A. In particular, this corresponds to be able to recognize what counts as a canonical proof of A and to make sure that the inference rules used in this proof are correct, i.e. valid. The correctness of the rules is usually assured by the inversion principle we mentioned in §7.1: what can be drawn from the elimination rules of a certain connective must already be drawn from the premisses of its corresponding introduction rules. As Sundholm (2004, p. 454) remarked, the peculiarity of this principle is that it «[. . .] leads straightforwardly to a resurrection of the old idea that the validity of an inference resides in the analytic containment of the conclusion in the premisses ». It is this very possibility of reducing proofs to "analytic proofs" that plays a crucial role in guaranteeing the respect of a molecularist approach: from the syntactical form of a given complex sentence A it is possible to extract a relevant information which allows to put a bound on the set of sentences necessary to have a full understand of A. In particular, if the inversion principle is respected, it could be possible to prove something like the subformula property, which is a property guaranteeing that if a complex sentence A is provable, then there exists a proof the rules of which are applied only on subformulas of the conclusion.[38]

But if we now look at the computational approach, we can see that in spite of some analogies with the Dummettian approach, crucial differences still hold. First, it should be noticed that in analogy with the Dummettian perspective, the understanding of the meaning of a sentence A is reduced to the understanding of the principal connective of A so that, eventually, it could be thought that it is sufficient to consider only a fragment of the language, namely the set of sentences in which this connective is principal. However, differently form the verificationist approach, the computational perspective presented here does not assume that the introduction rules and the harmony requirement are *per se* meaning-

[38] In fact, sometimes it could already be sufficient to prove a weaker property, like the subterm property. There are some theories - like the theory of equality, of groupoides and of lattices - for which the fact that a proof of A can make appeal to no other terms than those appearing in A is already sufficient to impose a bound on the set of formulas that should be known in order to know the meaning of A. The reason is that, from a technical point of view, for these theories the subterm property works like the subformula property: it allows to define proof-search methods by limiting the proof-search space (cf. Negri & von Plato 2011, §4).

conferring entities with respect to a certain connective. The reason is that, in order to know the meaning of A – or better, the meaning of the principal connective of A – it is not asked to know how to directly justify the assertion of A, but it is asked instead to know how to construct an argument against A. What counts then is to have a strategy in order to be able to refute A or – which is the same – a strategy in favor of $\neg A$. Thus, the meaning-conferring units are represented by argumentative strategies in their whole and not by single inference rules. The only property that these strategies are required to possess is to terminate when facing an argument in favor of A, no other properties are needed. In particular, it is not *a priori* required that the argument for $\neg A$ possess a particular order of the inference rules, neither that all the applied rules are correct (i.e. valid). This means that nothing like analyticity constraints are imposed on these arguments. What really matters is the strategy that has to be followed in order to refute A, while the focus on the formulas used in applying this strategy takes a backseat. The consequence is that there is no *a priori* limitations on the formulas involved in the arguments used for refuting A, and this could be thought to lead towards an holistic account of the computational approach we presented in this article.

However, it may be wondered whether this interpretation does not eventually conflict with what has been said about the feasibility properties – and thus with the existence of some kind of internal limitations – that seem to characterize the computational approach. Still, our opinion is that there is no real contradiction between these two aspects. Actually, what was under analysis in §7.2 was the fact that the knowledge of the meaning of a sentence is *manifested* by a certain speaker in an effective and concrete way, namely by operating a linguistic exchange with another speaker and this exchange involving only a bounded amount of resources. On the contrary, what we are analyzing here is the fact that it cannot be established in advance which is the fragment of the language that the speaker has to master in order to perform such linguistic exchanges with other speakers. This fact is not particularly astonishing. As we already said, the computational perspective goes along with a descriptive understanding of what a theory meaning is. From this perspective, what counts is to point out the competences that are manifested by those speakers who are able to participate to linguistic exchanges, but there is no pretension to fix these competences in advance, and especially there is no interest in trying to explain which are the characteristic features that a language must possess in order to be learnable.

On the contrary, focusing on the molecularity property reveals that a special attention is directed towards the problem of the *learning* of a language (see Dummett 1993, p. ix). In particular, in reason of their limited cognitive capacities, human agents can process only a limited amount of information at a time, so that in order to learn the meaning of an expression, it is necessary to master at most a finite fragment of the language (see Dummett 1973a, p. 515), otherwise it would go beyond human capacities, which are by hypothesis finite ones.

8 Conclusion

In the present work, we tackled the problem of explaining the meaning of mathematical or, in general, proper axioms, without appealing neither to a primitive notion of truth nor to other realist assumptions. Broadly speaking, our approach can thus be characterized as an anti-realist one.

Nevertheless, a major difficulty arises when one tries to interpret the notion of axiom through a Dummettian verificationist anti-realist semantics. The addition of axioms to standard proof systems entails the loss of [the notion of] canonical proofs, which is in fact the cornerstone of a verificationist theory of meaning. The existing solutions to this problem require a deep change in the epistemological status of axioms: axioms are turned into specific kind of rules. As a result, these solutions lead to revisionist positions with respect to the architecture of mathematical theories. In order to partly overcome these difficulties, we embraced a computational position. From such a perspective, we have been able to enlarge the set of primitive semantic concepts of the underlying theory of meaning, in a way still compatible with an inferentialist anti-realist approach.

Our strategy was twofold. First, we explored the computational aspects of a proof, considered as an "isolated" object, *via* proof-search algorithmic techniques. A careful analysis of the occurrences of the ✠ rule allowed us to show a precise correspondence between (logically incorrect) generalized axioms rules and "counter-models" using a homogenous proof-theoretical setting. Secondly, we explored the computational aspects of the interaction between proofs, considered as objects interacting through the Cut rule. This approach allows to "forget" formulas, by focusing only on the geometry of rules and their interactions. In this setting, generalized axioms provide a characterization of the crucial notion of paraproof. Both of these computational viewpoints reinforce the idea that generalized axioms are a way of working with (counter-)models inside the syntax. Axioms can thus be seen as fundamental "objects" at the crossroad between the syntax and the semantics of proof systems.

In the final part of the paper we made explicit some philosophical assumptions allowing us to integrate the analysis of untyped proofs with an inferentialist theory of meaning. We carried out such an analysis by pointing out some crucial differences between the inferentialist account based on Dummettian verificationism and the one based on the interactional approach presented in §5. Despite the fact that both accounts do not consider the notion of truth as primitive but as epistemically dependent, still a major divergence exists between them. It amounts to the difference between a normative (and "solipsistic") theory of meaning and a descriptive (and "social") one. In the end, we showed in which sense the shift from the former to the latter leads to embrace an even more radical form of anti-realism. Finally, we concluded our work with a short discussion around holism and molecularity. In order to have a full comprehension of the theory of meaning standing behind the computational and interactional approach presented in this paper, it seems to us unavoidable to establish whether this theory of meaning goes with an holistic or a molecularistic approach. The answer to this question is not yet established and seems to us valuable of further research.

References

BARNES, J. (1993). "Commentary to Aristotle, *Posterior Analytics*, pp. 81-271. Oxford: Clarendon Press.

BERNAYS, P. (1967). "David Hilbert". In P. Edwards (Ed.), *Encyclopedia of Philosophy*, Vol. 3, pp. 496-505. New York : MacMillan.

BOLZANO, B. (1817). *Rein analytischer Beweis des Lehrsatzes, dass zwischen je zwey Werthen, die ein entgegengesetztes Resultat gewähren, wenigstens eine reelle Wurzel der Gleichung liege*. Prague: Gottlieb Haase.

BONNAY, D. (2002). *Le contenu computationnel des preuves:* No-counterexemple interpretation *et spécification des théorèmes de l'arithmétique*. Master thesis, Université Paris 7 Paris Diderot.

BONNAY, D. (2007). "Règles et signification: le point de vue de la logique classique". In J.-B. Joinet (Ed.), *Logique, dynamique et cognition*, pp. 213-231. Paris: Publications de la Sorbonne.

BROUWER, L.E.J. (1908). "De Onbetrouwbaarheid der logische principes", Eng. trans. "The unreliability of the logical principles", in A. Heyting (Ed.), *L.E.J. Brouwer Collected Works : Philosophy and foundations of mathematics*,Vol. 1, pp. 443-446. Amsterdam: North-Holland, 1974.

CELLUCCI, C. (2004). "Introduzione". In B. Bolzano, *Del metodo matematico*, It. trans. L. Giotti, p. 7-39. Torino: Bollati Boringhieri.

DI COSMO, R. & MILLER, D. (2010). "Linear logic". In E.N. Zalta (Ed.), *Stanford Encyclopedia of Philosophy*, <http://plato.stanford.edu/archives/fall2010/entries/logic-linear>.

DOWEK, G. (2010). "From proof theory to theories theory", manuscript.

DOWEK, G., HARDIN, T. & KIRCHNER, C. (2003). "Theorem proving modulo". *Journal of Automated Reasoning*, 31 (1): 33-72.

DUBUCS, J. (2002). "Feasibility in logic". *Synthese*, 132 (3): 213-237.

DUBUCS, J. (2003). "Preuves, fondements et certificats". *Philosophia Scientiæ*, 7 (1): 167-198.

DUBUCS, J. & MARION, M. (2003). "Radical anti-realism and substructural logics". In A. Rojszczak, J. Cachro and G. Kurczewski (Eds.), *Philosophical Dimensions of Logic and Science. Selected Contributed Papers from the 11th International Congress of Logic, Methodology, and the Philosophy of Science, Kraków*, pp. 235-249. Dordrecht: Kluwer.

DUMMETT, M. (1973a). *Frege: Philosophy of Language*. London: Duckworth.

DUMMETT, M. (1973b). "The philosophical basis of intuitionistic logic". In Id., *Truth and Other Enigmas*, pp. 215-247. London: Duckworth, 1978.

DUMMETT, M. (1976). "What is a theory of meaning? (II)". In G. Evans and J. McDowell (Eds.), *Truth and Meaning: Essays in Semantics*, pp. 67-137. Oxford : Clarendon Press.

DUMMETT, M. (1977). *Elements of Intuitionism*. Oxford: Clarendon Press.

DUMMETT, M. (1991). *The Logical Basis of Metaphysics*. London: Duckworth.

DUMMETT, M. (1993). *The Seas of Language*. Oxford: Clarendon Press.

GENZTEN, G. (1934-35). "Untersuchungen über das logische Schliessen", Eng. trans. "Investigations into logical deduction", in M.E. Szabo (Ed.), *The Collected Papers of Gerhard Gentzen*, pp. 68-131. Amsterdam: North-Holland, 1969.

GENTZEN, G. (1938). "Neue Fassung des Widerspruchsfreiheitsbeweises für die reine Zahlentheorie", Eng. trans. "New version of the consistency proof for elementary number theory", in M.E. Szabo (Ed.), *The Collected Papers of Gerhard Gentzen*, pp. 252-308. Amsterdam: North-Holland, 1969.

GIRARD, J.-Y. (1987a). *Proof Theory and Logical Complexity*, Vol. 1. Naples: Bibliopolis.

GIRARD, J.-Y. (1987b). "Linear logic". *Theoretical Computer Science*, 50 (1): 1-101.

GIRARD, J.-Y. (2001). "Locus Solum: From the rules of logic to the logic of rules". *Mathematical Structures in Computer Science*, 11 (3): 301–506.

GIRARD, J.-Y. (2011). *The Blind Spot*. Zürich: European Mathematical Society Publishing.

GIRARD, J.-Y., LAFONT, Y. & TAYLOR, P. (1989). *Proofs and Types*. Cambridge: Cambridge University Press.

HALLETT, M. (1995). "Hilbert and logic". In M. Marion et S. Cohen (Eds.), *Québec Studies in Philosophy of Science*, 1, 135-187. Dordrecht: Kluwer.

HEYTING, A. (1962). "Axiomatic method and intuitionism". In Y. Bar-Hillel (Ed.), *Essays on the Foundations of Mathematics: Dedicated to A.A. Fraenkel on his Seventieth Anniversary*, pp. 237-247. Jerusalem: Magnes Press.

HILBERT, D. (1905). "Logische Principien des mathematischen Denkens", Ms. Vorlesung SS 1905, annotated by E. Hellinger, Bibliothek des Mathematischen Seminars, Universität Göttingen.

HILBERT, D. (1926). "Über das Unendliche", Eng. trans. "On the infinite", in P. Benacerraf and H. Putnam (Eds.), *Philosophy of Mathematics: Selected writings* (2nd edition), pp. 183-201, Cambridge: Cambridge University Press, 1983.

HILBERT, D. & BERNAYS, P. (1934). *Grundlagen der Mathematik*, Vol. 1. Berlin: Springer.

HINDLEY, J.R. & SELDIN, J.P. (2008). *Lambda-Calculus and Combinators: An introduction*. Cambridge: Cambridge University Press.

HINTIKKA, J. (1983). *The Game of Language: Studies in game-theoretical semantics and its applications*, in collaboration with J. Kulas. Dordrecht: Kluwer.

HINTIKKA, J. (2011). "What is the axiomatic method?". *Synthese*, 183 (1): 69-85.

KLEENE, S.C. (1945). "On the interpretation if intuitionistic number theory". *The Journal of Symbolic Logic*, 10 (4): 109-124.

KLEENE, S.C. (1973). "Realizability: A retrospective survey". *Cambridge Summer School in Mathematical Logic*, Lecture Notes in Mathematics, 337: 95-112.

KREISEL, G. (1951). "On the interpretation of non-finitist proofs. Part I". *The Journal of Symbolic Logic*, 16 (4): 241-267.

KREISEL, G. (1952). "On the interpretation of non-finitist proofs: Part II. Interpretation of number theory". *The Journal of Symbolic Logic*, 17 (1): 43-58.

KREISEL, G. (1960). "Ordinal logics and the characterization of informal notions of proof". In J.A. Todd (Ed.), *Proceedings of the International Congress of Mathematicians. Edinburgh, 14-21 August 1958*, pp. 289-299. Cambridge: Cambridge University Press.

KRIVINE, J.-L. (2003). "Dependent choice, 'quote' and the clock". *Theoretical Computer Science*, 308 (1-3): 259-276.

LECOMTE, A. & QUATRINI, M. (2011a). "Figures of dialogue: A view from ludics". *Synthese*, 183 (1) Supplement: 279-305.

LECOMTE, A. & QUATRINI, M. (2011b). "Ludics and rhetorics". In A. Lecomte et S. Tronçon (Eds.), *Ludics, Dialogue and Interaction. PRELUDE Project 2006-2009: Revised selected papers*, Lecture Notes in Artificial Intelligence, Vol. 6505, pp. 32-59. Berlin-Heidelberg-New York: Springer.

LORENZEN, P. & LORENZ, K. (1978). *Dialogische Logik*. Darmstadt: Wissenschaftliche Buchgesellschaft.

MARION, M. (2009). "Radical anti-realism, Wittgenstein and the length of proofs". *Synthese*, 171 (3): 419-432.

MARTIN-LÖF, P. (1970). *Notes on Constructive Mathematics*. Stockholm: Almqvist & Wiksell.

MARTIN-LÖF, P. (1987). "Truth of a proposition, evidence of a judgment, validity of a proof". *Synthese*, 73 (3): 407-420.

MARTIN-LÖF, P. (1996). "On the meanings of the logical constants and the justifications of the logical laws". *Nordic Journal of Philosophical Logic*, 1 (1): 11-60.

MIQUEL, A. (2009a). *De la formalisation des preuves à l'extraction de programmes*. Habilitation thesis, Université Paris 7 Paris Diderot.

MIQUEL, A. (2009b). " Classical realizability with forcing and the axiom of countable choice", manuscript.

NAIBO, A. (2013). *Le statut dynamique des axiomes. Des preuves aux modèles*. PhD thesis, Université Paris 1 Panthéon-Sorbonne.

NEGRI, S. & VON PLATO, J. (1998). "Cut elimination in the presence of axioms". *The Bulletin of Symbolic Logic*, 4 (4): 418-435.

NEGRI, S. & VON PLATO, J. (2001). *Structural Proof Theory*. Cambridge: Cambridge University Press.

NEGRI, S. & VON PLATO, J. (2011). *Proof Analysis: A Contribution to Hilbert's Last Problem*. Cambridge: Cambridge University Press.

PAOLI, F. (2002). *Substructural Logics: A primer*. Dordrecht: Kluwer.

PARSONS, C. (2008). *Mathematical Thought and Its Objects*. Cambridge: Cambridge University Press.

VON PLATO, J. (2003). "Translations from natural deduction to sequent calculus". *Mathematical Logic Quarterly*, 49 (5): 435-443.

VON PLATO, J. (2007). "In the shadows of the Löwenheim-Skolem theorem: Early combinatorial analyses of mathematical proofs". *The Bulletin of Symbolic Logic*, 13 (2): 189-225.

PRAWITZ, D. (1965). *Natural Deduction: A proof-theoretical study*. Stockholm: Almqvist & Wiksell.

PRAWITZ, D. (1973). "Towards a foundation of a general proof theory". In P. Suppes et al. (Eds.), *Logic, Methodology and Philosophy of Science IV. Proceedings of the Fourth International Congress for Logic, Methodology and Philosophy of Science. Bucharest, 1971*, pp. 225-250. Amsterdam: North-Holland.

PRAWITZ, D. (2012a). "The epistemic significance of valid inference". *Synthese*, 187 (3): 887-898.

PRAWITZ, D. (2012b). "Truth as an epistemic notion". *Topoi*, 31 (1): 9-16.

SCHÜTTE, K. (1956). "Ein System des verknüpfenden Schliessens". *Archiv für mathematische Logic und Grundlagenforschung*, 2 (2-4): 55-67.

SEILLER, T. (2012a). "Interaction graphs: Additives", arXiv:1205.6557.

SEILLER, T. (2012b). "Interaction graphs: Multiplicatives". *Annals of Pure and Applied Logic*, 163 (12): 1808-1837.

SEILLER, T. (2012c). *Logique dans le facteur hyperfini: Géométrie de l'interaction et complexité*. PhD thesis, Université Aix-Marseille.

SEILLER, T. (2013). "Interaction graphs: Exponentials", arXiv:1312.1094.

SØRENSEN, M.H. & URZYCZYN, P. (2006). *Lectures on the Curry-Howard Isomorphism*. Amsterdam: Elsevier.

SUNDHOLM, G. (1983). "Constructions, proofs and the meaning of logical constants". *Journal of Philosophical Logic*, 12 (2): 151-172.

SUNDHOLM, G. (1994). “Vestiges of realism”. Dans B. McGuinness et G. Oliveri (éds.), *The Philosophy of Michael Dummett*, pp. 137-165. Dordrecht: Kluwer.

SUNDHOLM, G. (1997). “Implicit epistemic aspects of constructive logic”. *Journal of Logic, Language, and Information*, 6 (2): 191-212.

SUNDHOLM, G. (1998). “Proofs as acts and proofs as objects: Some questions for Dag Prawitz”.*Theoria*, 64 (2-3): 187-216.

SUNDHOLM, G. (2004). “Antirealism and the roles of truth”. Dans I. Niiniluoto, M. Simonen et J. Woleński (éds.), *Handbook of Epistemology*, pp. 437-466. Dordrecht: Kluwer.

TENNANT, N. (2012). “Inferentialism, logicism, harmony, and a counterpoint”. In A. Miller (Ed.), *Essays for Crispin Wright: Logic, language and mathematics*, Vol. 2. Oxford: Oxford University Press, to appear.

TROELSTRA, A.S. & VAN DALEN, D. (1988). *Constructivism in Mathematics*, Vol. 2. Amsterdam: North-Holland.

TROELSTRA, A.S. & SCHWICHTENBERG, H. (2000). *Basic Proof Theory* (2nd edition). Cambridge: Cambridge University Press.

WANG, H. (1981). *Popular Lectures on Mathematical Logic*. New York: van Nostrand.

WITTGENSTEIN, L. (1956). *Bemerkungen über die Grundlagen der Mathematik*, edited by G.H. von Wright, R. Rhess and G.E.M. Anscombe, translated by G.E.M. Anscombe, *Remarks on the Foundations of Mathematics*. Oxford: Basil Blackwell.