Introduction

"The truth, the whole truth, and nothing but the truth." This famous oath could have constituted, back in the 17th century, Leibniz's profession of faith in seek of his *calculus ratiocinator*. Indeed he envisioned that every philosophical dispute may be settled by a calculation [107][1]:

> *"The only way to rectify our reasonings is to make them as tangible as those of the Mathematicians, so that we can find our error at a glance, and when there are disputes among persons, we can simply say: Let us calculate [calculemus], without further ado, to see who is right. […] if controversies were to arise, there would be no more need of disputation between two philosophers than between two calculators. For it would suffice for them to take their pencils in their hands and to sit down at the abacus, and say to each other […]: Let us calculate"*

While there are reasonable doubts about whether Leibniz intended for the so-called *calculus ratiocinator*, the system or device used to perform these logical deductions, to be an actual machine[2] or simply an abstract calculus, it is certain that he hoped to reduce all human reasonings to computation.

Alas, an obstacle—and not the least— was standing on his way: at the time, reasoning was taking the form of informal text, even in mathematics. Leibniz was then about to initiate a long path towards the formalization of mathematics. As a first step, he proposed the concept of *characteristica universalis* which was meant to embody every human concept. Leibniz indeed had a combinatorial view of human ideas, thinking that they *"can be resolved into a few as their primitives"* [106, p. 205]. This idealistic language should thereby assign a character to each primitive concept, from which we could form characters for derivative concepts by means of combinations of the symbols: *"it would be possible to find correct definitions and values and, hence, also the properties which are demonstrably implied in the definitions"* [106, p. 205]. Leibniz thus intended for the *characteristica universalis* to be a universal language, which was to be employed in the computation of the calculus ratiocinator. If, at the end of the story, this dream turned out to be a chimera, we should acknowledge that his set idea of relating logic to computation was brightly visionary. Due do this connection, we can trumpet that this thesis is part of a tradition of logic initiated by Leibniz himself. To find our way back from the present dissertation to the *calculus ratiocinator*, let us identify a few milestones[3] along the path.

It actually took two centuries until a major step was made in direction of a formalization of mathematics. In the meantime, the scientific community had to handle an episode which shook the very foundations of mathematics: the discovery of non-Euclidean geometries. Two millenia earlier, Euclid gave in his *Elements* the first axiomatic presentation of geometry. He placed at the head of his treatise a collection of definitions (*e.g. "a line is a length without breadth"*), common notions (*e.g. "things equal to*

[1] For the reader looking for a good old exercise of Latin, here comes the original quote [106, p. 200]: *Quo facto, quando orientur controversiae, non magis disputatione opus erit inter duos philosophus, quam inter duos computistas. Sufficiet enim calamos in manus sumere sedereque ad abacos, et sibi mutuo (accito si placet amico) dicere: calculemus.*

[2] Leibniz was one of the pioneers of mechanical calculator with his Stepped Reckoner, the first machine with all four arithmetic abilities.

[3] Amusingly, *calculus* precisely means *stone* in Latin. Despite serious *scrupula*, we could not refrain ourselves from annoying the reader with this insignificant observation.

the same thing are also equal to one another") and five postulates (*e.g. "to draw a straight-line from any point to any point"*). Amongst these postulates, the fifth, also called the *parallel postulate*, has literally retained mathematicians' attention for a thousand years:

> *If a straight line crossing two straight lines makes the interior angles on the same side less than two right angles, the two straight lines, if extended indefinitely, meet on that side on which are the angles less than the two right angles.*

Because of its surprising prolixity with respect to the first four postulates, numerous attempts were made with the aim of deducing the parallel postulate from the first four, none of them showed to be successful. In the 1820s, Nikolai Lobachevsky and János Bolyai independently tackled the problem in a radically new way. Instead of trying to obtain a proof of the parallel postulate, Bolyai considered a theory relying only on the first four postulates, which he called "absolute geometry" [19], leaving the door open to a further specification of the parallel postulate or its negation. In turn, Lobatchevsky built on the negation of the parallel postulate a different geometry that he called "imaginary" [110]. Interestingly, to justify the consistency of his system, Lobachevsky argued that any contradiction arising in his geometry would inevitably be matched by a contradiction in Euclidean geometry. This appears to be the earliest attempt of a proof of relative consistency[4]. A few years later, Bernhard Riemman published a dissertation in which he also constructed a geometry without the parallel postulate [145]. For the first time, some mathematical theories were neither relying on synthetic *a priori* judgments nor on empirical observations, and yet, they were consistent in appearance. Theses new geometries, by denying traditional geometry its best claim to certainty, posed to the community of mathematicians a novel challenge: *How can it be determined for sure that a theory is not contradictory?* If Leibniz was our first milestone on the way, we would like the second one to mark this question.

For years, non-Euclideans geometries have been the target of virulent criticism, the colorful language of which the decency forbids us from transcribing here. One of the strongest opponent to these geometries was Gottlob Frege, who notably wrote: *"No man can serve two masters. One cannot serve the truth and the untruth. If Euclidean geometry is true, then non-Euclidean geometry is false."* [49]. Frege was thus in line with the ground postulate of Leibniz's calculus ratiocinator that the truth of any statement can be decided. In this perspective, Frege accomplished a huge step for the formalization of mathematics. In 1879, he introduced his *Begriffsschift* [48], a formal language to express formulas and proofs. Frege aimed at expressing abstract logic by written signs in a more precise and clear manner than it would be possible by words (which is not without recalling Leibniz's intentions with the characteristica universalis). Especially, Frege was responsible for the introduction of the quantifiers $\forall$—"*for all*"—and $\exists$—"*there exists*"—and most importantly of a proof system based on axioms and inference rules. Thereby, he paved the way for a syntactic study of proofs, emphasizing the *provability* of formulas.

On the other hand, the earlier work of Boole [20] did not lead to a language peculiar to logical considerations, but rather to the application of the laws (and symbols) of algebra[5] to the realm of logic. In particular, Boole's approach consists in assigning a truth value to each proposition, pointing out the semantic notion of *validity* of formulas.

Despite Boole and Frege advances, when the 20$^{\text{th}}$ century began, the existence of *calculus ratiocinator* was still a plausible expectation in light of the state of the art in logic. Even without matching

[4] Actually, there is an earlier trace of such a proof in Thomas Reid's work [142]. He defined a non-Euclidean geometry, his so-called *"geometry of visibles"*, that he described as being the one perceived by the *Idomenians*, some imaginary beings deprived of the notion of thickness. Reid claims that the *"visible"* space can be represented by an arbitrary sphere encompassing the space. This can also be considered as a relative consistency proof, asserting that the geometry of visibles is consistent if spherical geometry is. A detailed discussion on Reid's geometry can be found in [36].

[5] According to Boole, *"the operations of Language, as an instrument of reasoning, may be conducted by a system of signs composed of [...] literal symbols $x, y, \ldots$ [...] signs of operation, as $=$, $-$, $\times$ [...] the sign of identity $=$. And these symbols of Logic are in their use subject to definite laws, partly agreeing with and partly differing from the laws of the corresponding symbols in the science of Algebra"* [20, Chapter II].

Leibniz's ambition of deciding the validity of any philosophical statement, the problem of deciding the truth merely within mathematics was still an open question. In 1900, Hilbert drew up a list of twenty-three problems—another milestone along our travel time—the second of which was to prove the compatibility of the arithmetical axioms, *"that is, that a finite number of logical steps based upon them can never lead to contradictory results"* [73]. Rooted in this question, Hilbert established in the 1920s a program aiming at a formalization of all mathematics in axiomatic form, together with a proof that this axiomatization is consistent. Hilbert's manifesto for a quest of foundations climaxed with the slogan *"No ignorabimus"* during a radio broadcast in 1930[6] [74]:

> *"For us mathematicians, there is no 'ignorabimus', and, in my opinion, there is none whatsoever for the natural sciences. In place of this foolish 'ignorabimus' let our watchword on the contrary be: We must know — we shall know!"*

In continuation of his program, Hilbert raised with Ackermann another fundamental question in 1928, which is known as the *Entscheidungsproblem* [75]: to decide if a formula of first-order logic is a tautology. By "to decide" is meant via an algorithm, by means of a procedure. The signification of "algorithm" should be taken in context: the very concept of computer was yet unknown, an algorithm was thus to be understood as a methodical way of solving a problem, as a computational recipe. By putting the computation at the heart of the problem, the *Entscheidungsproblem* enters directly into the heritage of Leibniz quest for a calculus ratiocinator.

Unfortunately, Hilbert's fine aspirations were quickly shattered. First by Gödel [61], who proved in 1931 that any consistent logical system, provided that it is expressive enough, featured a formula which is not provable in this system, nor is its negation. Worst, he showed in particular that the consistency of arithmetic could not be proved within arithmetic, giving then a definitive and negative answer to Hilbert's second problem. As for the *Entscheidungsproblem*, Church [25, 26] and Turing [153, 154] independently proved that no algorithm could ever decide the validity of first-order formulas. Both answers relied on a specific definition of the notion of computability, captured in one case by *Turing machines*, by the *λ-calculus* in the case of Church. Church and Turing proved that both formalisms were equivalent, laying the ground of a unified definition of what are the *"computable"* functions. In other words, the concept of computer was born.

Leaving aside a few decades and some noteworthy discoveries, the second to last milestone on our journey, arguably the most important one concerning this thesis, is due to Curry [33, 34] and Howard [77], in 1934 and 1969 respectively. Independently, they both observed that the proofs of a constructive subset of mathematics, called intuitionistic logic, coincide exactly with a typed subset of the λ-calculus. This observation had a particularly significant consequence: by asserting that (intuitionistic) *proofs* were nothing less than *programs*, it put the computation at the center of modern proof theory. Furthermore, it brought kind of a small revolution by giving the possibility of designing altogether a proof system and a programming language, bug-free by essence.

While the proofs-as-programs correspondence seemed for a time to be bounded to intuitionistic logic and purely functional programming language, Griffin discovered in 1990 that Scheme's control operator `call/cc` could be typed by a non-constructive principle named the law of Peirce [62]. Several calculi were born from this somewhat accidental breakthrough, allowing for a direct computational interpretation of classical logic. Especially, Krivine developed the theory of *classical realizability* based on an extension of the λ-calculus with `call/cc`, in which he tried to obtain programs for well-known axioms. In so doing, he adopted a conquerent state-of-mind, proposing to push further the limits of Curry-Howard correspondence by programming new proofs.

[6] In case some readers would not have found satisfaction with the former Latin exercise, here his the original German declaration: *"Für uns gibt es kein Ignorabimus, und, meiner Meinung nach, auch für die Naturwissenschaft überhaupt nicht. Statt des törichten Ignorabimus, heiße im Gegenteil unsere Losung: Wir müssen wissen — wir werden wissen!"*.

Yet, it would be unfair to reduce classical realizability, our last milestone, to its sole contribution to proof theory. To highlight its particular significance, allow us a slight digression back to the early 1900s. Indeed, we eluded in our presentation the fact that mathematics were affected by the so-called *foundational crisis*. To cut a long story short, Frege axiomatized in his *Begriffsschift* [48] a *set theory* built on Cantor's earlier ideas. This theory was intended to lay a foundational ground to the definition of all mathematics, but a few years later a paradox was discovered by Russell, proving the theory to be inconsistent. If the axiomatization of set theory was finally corrected by Zermelo and Fraenkel, further to this episode, the question of proving the consistency of a given axiomatization has been a central issue for logicians of the 20$^{\text{th}}$ century. Two axioms were particularly controversial, namely the *axiom of choice* and the *continuum hypothesis*. Relying on Boole's notion of validity, Gödel first proved in 1938 that both were consistent with Zermelo-Fraenkel set theory [66]. Cohen finally proved that these axioms were independent from set theory, by showing that their negations were also consistent with set theory. To this end, he developed the technique of *forcing* to construct specific models in which these axioms are not valid.

At the edge of the last decade, Krivine showed in an impressive series of papers [98, 99, 100, 101] that classical realizability also furnishes a surprising technique of model construction for classical theories. In particular, he proved that classical realizability subsumes forcing models, and even more, gives raise to unexpected models of set theories. Insofar as it opens the way for new perspectives in proof theory and in model theory, we can safely state that classical realizability plays an important role in the (modern) proofs-as-programs correspondence.

This thesis is in line with both facets of classical realizability. On the one hand, from the point of view of syntax and *provability*, we continue here a work started by Herbelin in 2012 [70] which provides a proof-as-program interpretation of classical arithmetic with dependent choice. Half of this thesis is devoted to proving the correctness of Herbelin's calculus, called dPA$^{\omega}$, which takes advantage of several extensions of the proofs-as-programs correspondence to interpret the axiom of dependent choice. We rephrase here Herbelin's approach in a slightly different calculus, dLPA$^{\omega}$, of which we analyze the different computational features separately. We finally prove the soundness of dLPA$^{\omega}$, which allows us to affirm:

> *Constructive proofs of the axioms of countable and dependent choices can be obtained in classical logic by reifying the choice functions into the stream of their values.*

On the other hand, from the viewpoint of semantics and *validity*, we pursue the algebraic analysis of the models induced by classical realizability, which was first undertaken by Streicher [150], Ferrer, Guillermo, Malherbe [44, 45, 43], and Frey [50, 51]. More recently Miquel [121] proposed to lay the algebraic foundation of classical realizability within new structures which he called *implicative algebras*. These structures are a generalization of Boolean algebras (the common ground of model theory) based, as the name suggests, on an internal law representing the implication. Notably, implicative algebras allow for the interpretation of both programs (*i.e.* proofs) and their types (*i.e.* formulas) in the same structure. In this thesis, we deal with two similar notions: *disjunctive algebras*, which rely on internal laws for the negation and the disjunction, and *conjunctive algebras*, centered on the negation and the conjunction. We show how these structures underly specific models induced by classical realizability, and how they relate to Miquel's implicative algebras. In particular, if this part of the thesis were to be reduced to a take-away message, we would like this message to be:

> *The algebraic analysis of the models that classical realizability induces can be done within simple structures, amongst which implicative algebras define the more general framework.*

The main contributions of this thesis can be stated as follows.

1. A realizability interpretation *à la* Krivine of the $\overline{\lambda}_{[lv\tau\star]}$-calculus [4], which is a call-by-need calculus with control and explicit stores. This interpretation provides us with a proof of normalization for this calculus. In addition, it leads us toward a typed continuation-and-store-passing style translation, which relies on the untyped translation given in [4]. We relate the store-passing style translation with Kripke forcing translations.
2. A classical sequent calculus with dependent types, which we call dL. While dependent types are known to misbehave in presence of classical logic, we soundly combine both by means of a syntactic restriction for dependent types. We show how the sequent calculus presentation brings additional difficulties, which we solve by making use of delimited continuations. In particular, we define a typed continuation-passing style translation carrying the dependencies.
3. A proofs-as-programs interpretation of classical arithmetic with dependent choice, which we dLPA$^\omega$. Our calculus is an adaptation of Herbelin's dPA$^\omega$ system, given in a sequent calculus presentation. Drawing on the techniques previously developed for the $\overline{\lambda}_{[lv\tau\star]}$-calculus and dL, we defined a realizability interpretation of dLPA$^\omega$. This implies in particular the soundness and the normalization of dLPA$^\omega$, properties which were not proved yet for dPA$^\omega$.
4. A Coq formalization of Miquel's implicative algebras [121]. Since implicative algebras aim, on a long-term perspective, at providing a foundational ground for the algebraic analysis of realizability models, I believe that having a Coq development supporting the theory is indeed an appreciable feature.
5. The definition and the study of disjunctive algebras. We show how these structures, which are similar to implicative structures, naturally arise from realizability models based on the decomposition of the implication $A \to B$ as $\neg A \vee B$. We study the intrinsic properties of disjunctive algebras, and we prove that they are particular cases of implicative algebras.
6. The notion of conjunctive algebra, which relies on the decomposition of the implication $A \to B$ as $\neg(A \wedge \neg B)$. We explain how these structures naturally underly the realizability interpretations of some specific call-by-value calculus. We then prove that any disjunctive algebra induces a conjunctive algebra by duality. The converse implication and the properties of conjunctive algebras are yet to be studied.

The thesis itself is broadly organized according to the contributions listed above. We give here a description of the different chapters which compose this manuscript.

The first part of this thesis consists of a preliminary introduction to the scientific topics involved in the thesis. We attempt to be as self-contained as possible, and in particular these chapters are there to introduce well-known definitions and illustrate techniques which are relevant to the later contributions. As such, experts in the field should feel free to skip this part, all the more as back references are made to these chapters when necessary.

In Chapter 1, we give a self-contained introduction to formal logic, and present the concepts of theory, proof, and model. We come back in details to the notions of provability and validity evoked in the introduction, which we illustrate with several examples. Hopefully, this chapter should be accessible to anyone with a scientific background.

In Chapter 2, we introduce the λ-calculus, which is the fundamental model of computation for the study of functional programming languages. We first present the untyped λ-calculus, and we focus on the key properties that are in play in the study of such a calculus. We then present the simply-typed λ-calculus and the proofs-as-programs correspondence. Once again, this chapter is meant to be accessible to curious non-specialists, which may understand here the second half of this thesis title.

In Chapter 3, we give a survey of Krivine's classical realizability. In particular, we introduce the λ_c-calculus with its abstract machine, and we give in details the definition of classical realizability. We

then present some of its standard applications, both as a tool to analyze the computational behavior of programs and as a technique of model construction.

In Chapter 4, we present Gentzen's sequent calculus. together with its computational counterpart, Curien and Herbelin's $\lambda\mu\tilde{\mu}$-calculus. We take advantage of this section to illustrate (on the call-by-name and call-by-name $\lambda\mu\tilde{\mu}$-calculi) the benefits of continuation-passing style translations and their relations with realizability interpretations *à la* Krivine. In particular, the expert reader might be interested in our observation that Danvy's methodology of semantic artifacts can be used to derive realizability interpretations.

The second part of this thesis is devoted to the study of a proof system allowing for the definition of a proof term for the axiom of dependent choice.

In Chapter 5, we give a comprehensive introduction to Herbelin's approach to the problem with dPA$^\omega$ [70]. We explain how the different computational features of dPA$^\omega$—namely dependent types, control operators and a co-inductive fixpoint which is lazily evaluated—are used to prove the axioms of countable and dependent choices. We then focus on the difficulties in proving the soundness of dPA$^\omega$, which are precisely related to the simultaneous presence of all these features. Finally, we present our approach to the problem, and the organization of the subsequent chapters.

In Chapter 6, we present a call-by-need calculus with control, the $\overline{\lambda}_{[lv\tau\star]}$-calculus. This calculus features explicit environments in which terms are lazily stored, which we use afterwards in dLPA$^\omega$. To prepare the later proof of normalization for dLPA$^\omega$, we prove the normalization of the $\overline{\lambda}_{[lv\tau\star]}$-calculus by means of a realizability interpretation. We also give a typed continuation-and-store passing style, whose computational content highlights the already known connection between global memory and forcing translations.

In Chapter 7, we introduce dL, a sequent calculus with control and dependent types. Here again, the underlying motivation is to pave the way for the further introduction of dLPA$^\omega$. Nonetheless, such a calculus is an interesting object in itself, which motivates our thorough presentation of the topic. We thus explain how control and dependent types can be soundly combined by means of a syntactic restriction of dependencies. We show how the challenge posed by the sequent calculus presentation can be solved thanks to the unexpected use of delimited continuations. The latter has the significant benefits of making the calculus suitable for a typed continuation-passing style carrying the dependencies.

Finally, in Chapter 8, we present dLPA$^\omega$, a calculus which soundly combines all the computational features of dPA$^\omega$ in a sequent calculus fashion. We give a realizability interpretation for dLPA$^\omega$, whose definition relies on the interpretations previously defined for the $\overline{\lambda}_{[lv\tau\star]}$-calculus and dL. We deduce from this interpretation the soundness and normalization of dLPA$^\omega$, the primary objectives of this part of the thesis.

The third part of the thesis is dedicated to the study of algebraic structures arising from the models that Krivine's classical realizability induces.

In Chapter 9, we give a detailed introduction to the topic, starting from Kleene intuitionistic realizability to eventually reach the notion of realizability triposes. In particular, we recall some standard definitions of the categorical analysis of logic. Then we present the algebraic approach to classical realizability and the structures that are involved.

In Chapter 10, we present Miquel's implicative algebras [121], which aim at providing a general algebraic framework for the study of classical realizability models. We first give a self-contained presentation of the underlying implicative structures. We then explain how these structures can be turned into models by means of separators. Finally, we show the construction of the associated triposes together with some criteria to determine whether the induced model amounts to a forcing construction.

In Chapter 11, we follow the rationale guiding the definition of implicative algebras to introduce the notion of disjunctive algebra. Our main goal in this chapter is to draw the comparison with the implicative case, and especially to justify that the latter provides a more general framework than disjunctive algebras. After studying the properties peculiar to disjunctive algebras, we eventually prove that they indeed are particular cases of implicative algebras.

Last, in Chapter 12, we attempt to follow the same process in order to define the notion of conjunctive algebra. If we succeed in proving that any disjunctive algebra give raise to a conjunctive algebra by duality (which is to be related with the well-known duality between call-by-name and call-by-value), we do not prove the converse implication. We conclude by saying a word on the perspectives and questions related to the algebraization of classical realizability.